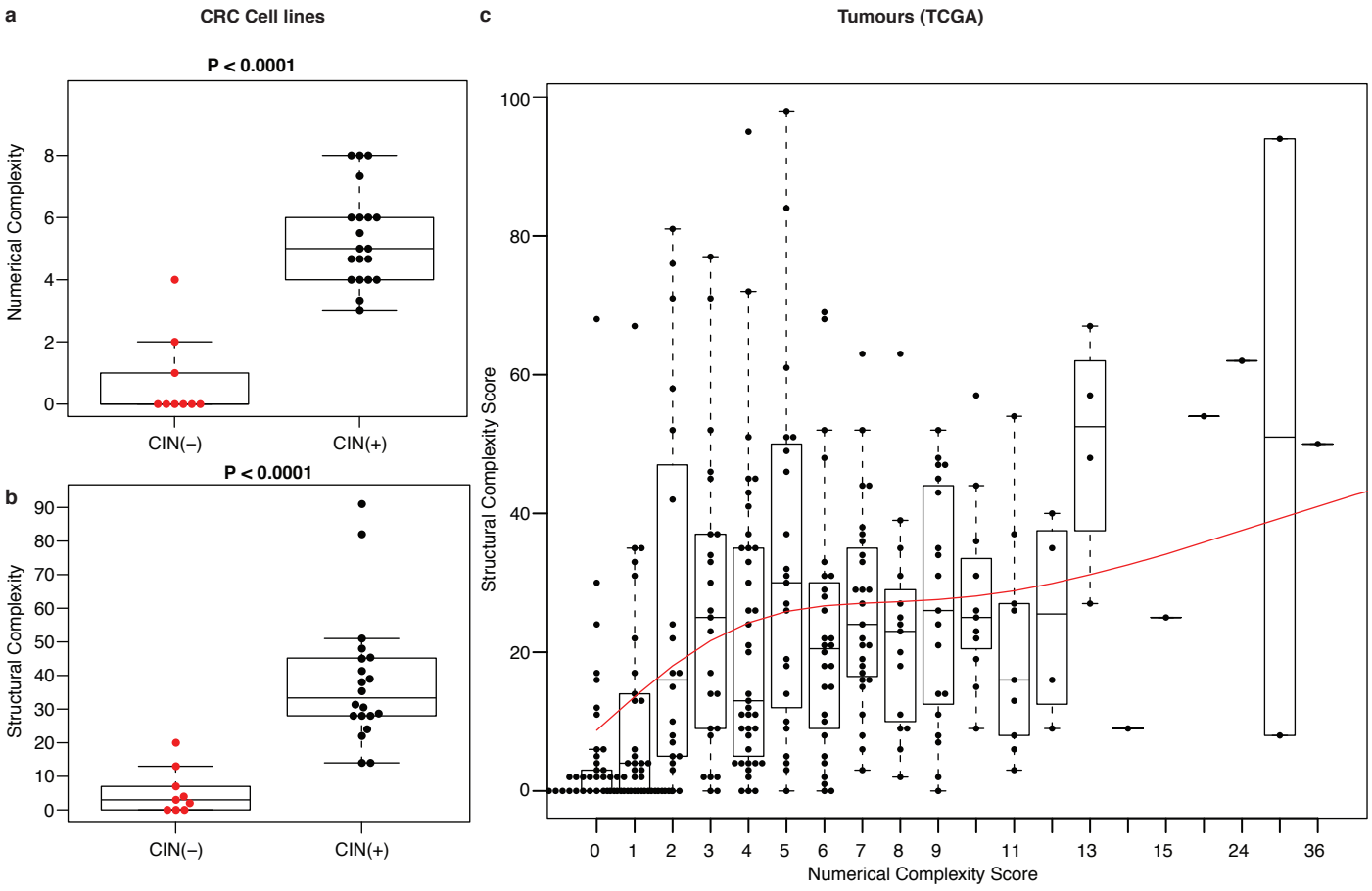
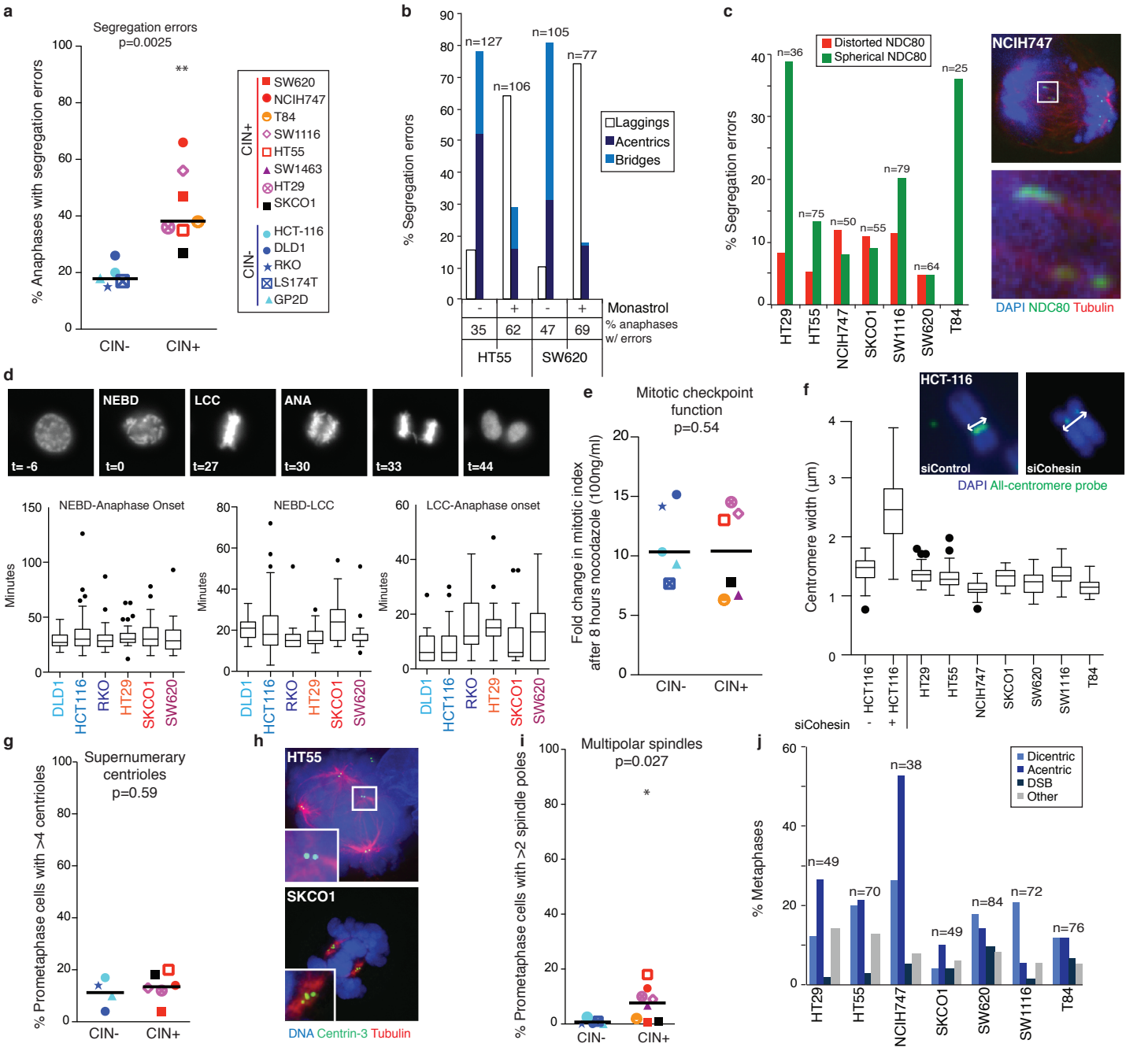
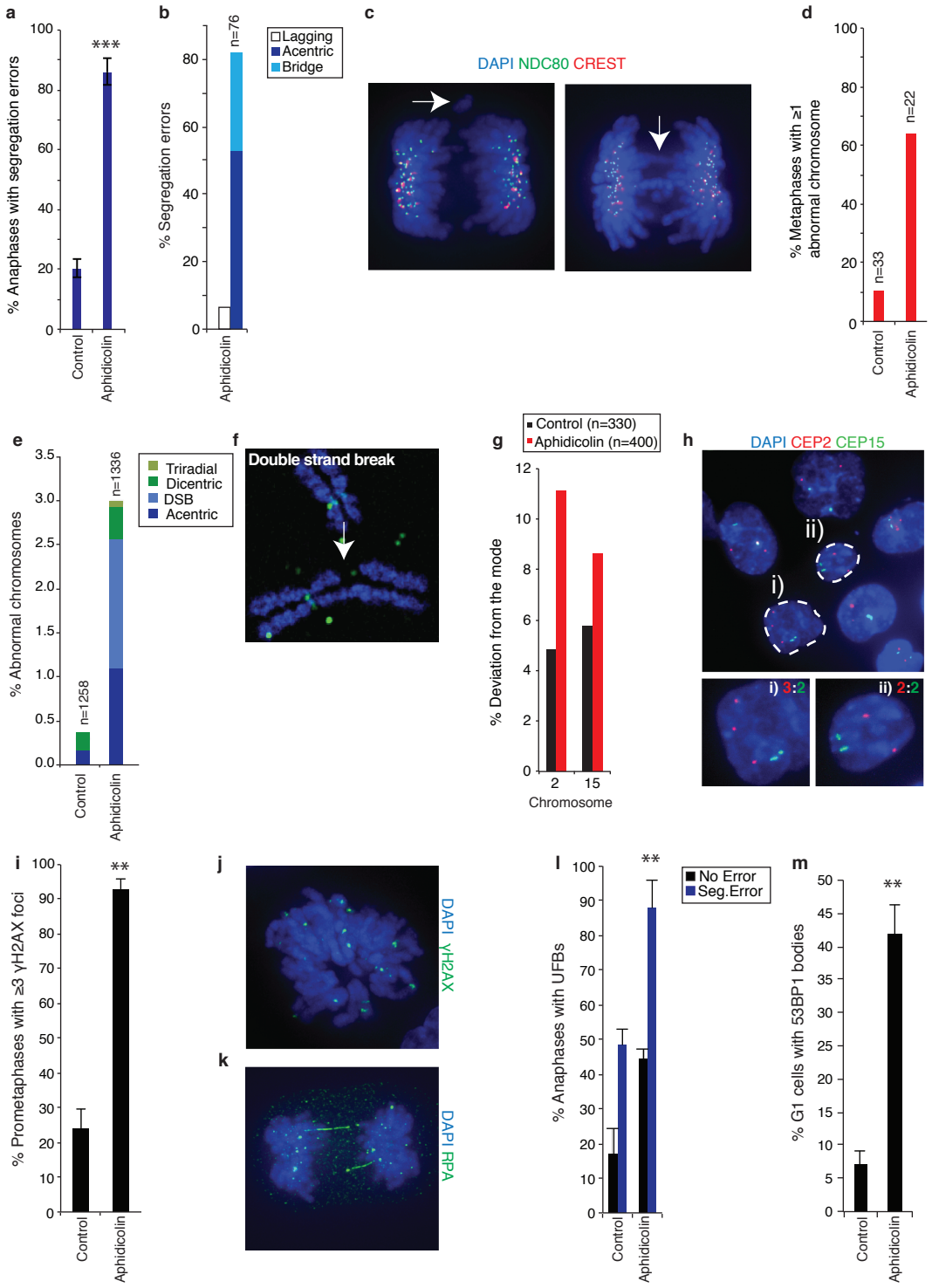


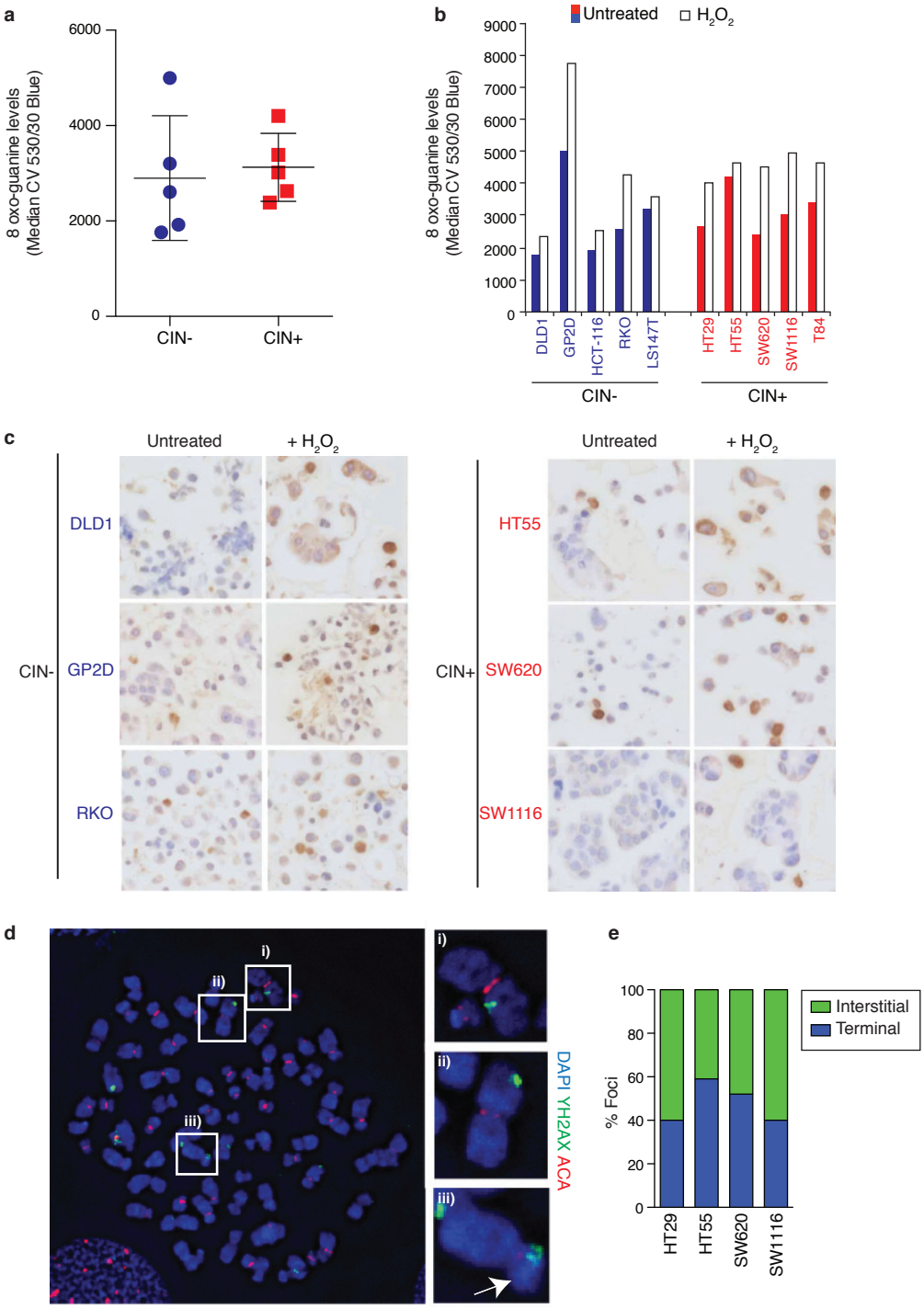
Supplementary Figure 1. Numerical and Structural CIN co-occur in CRC. a,b, Ploidy-normalised scores reflecting **(a)** numerical and **(b)** structural karyotypic complexity derived from SNP 6.0 data for CIN+ versus CIN- CRC cell lines (see Methods). **c,** Correlation between numerical and structural complexity in colorectal tumours (TCGA cohort[19], Spearman's rank correlation: r=0.4555, p < 0.0001).



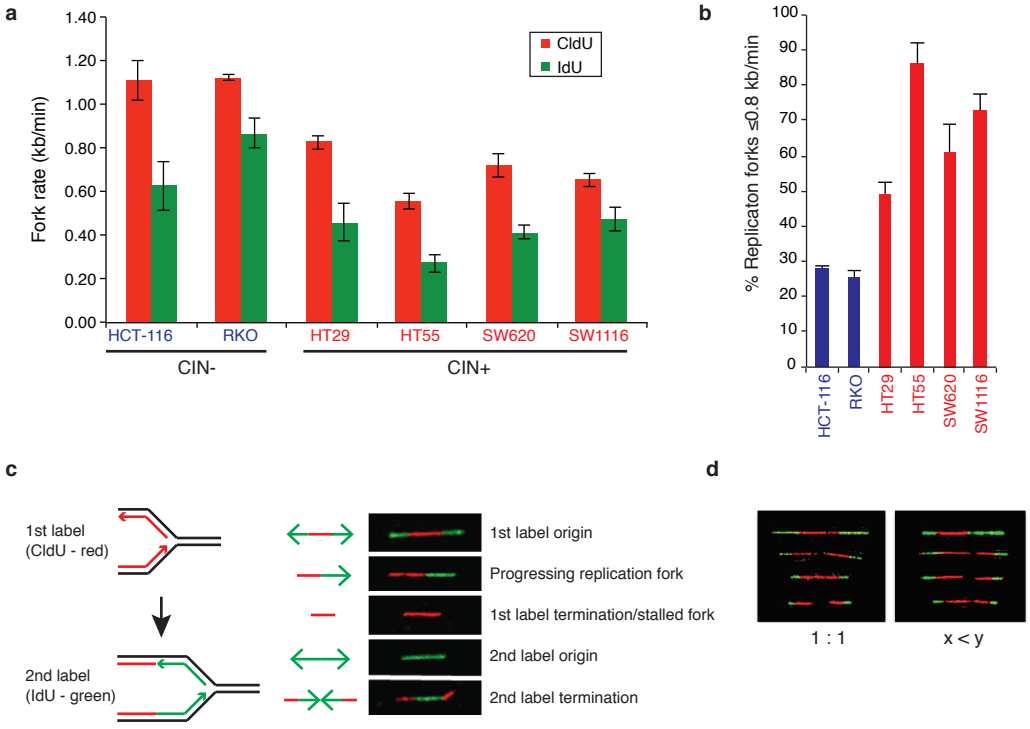
Supplementary Figure 2. Chromosome segregation errors in CIN+ CRC cells are independent of mitotic defects. a, % Anaphases exhibiting segregation errors (acentrics, bridges or lagging chromosomes) in a panel of CIN- versus CIN+ CRC cell lines (n=100 cells per cell line, black lines are median values, statistic: Mann-Whitney test p=0.0025). **b,** Classification of segregation errors in HT55 and SW620 cells ± 16 h monastrol (100 μM) to arrest cells in prometaphase with monopolar spindles, followed by drug washout for 75 min, elevating the frequency of improper chromosome attachments[6]. Monastrol washout resulted in specific induction of lagging chromosomes, not anaphase bridges or acentrics. **c,** CIN+ cells were stained with antibodies for NDC80, a microtubule binding component of the kinetochore, and β-tubulin to identify merotelic attachments. % Segregation errors in CIN+ cell lines that were lagging chromosomes with (red bars) or without (green bars) kinetochore distortion. **Top right:** Representative image of a merotelic attachment in an NCIH747 anaphase. NDC80 is distorted (**bottom right**). **d,** Cell lines stably expressing H2B-RFP were imaged every 3 minutes to assess duration of prometaphase (nuclear envelope breakdown (NEBD) to last chromosome congressed (LCC)), metaphase (LCC to anaphase onset), and the total duration of mitosis (NEBD to anaphase onset)(n=20-71 cells per cell line). Stills of SKCO1-H2B-mRFP (CIN+) cells illustrating these events are shown. **e,** To assess mitotic checkpoint proficiency, cells were incubated with nocodazole (100 ng/ml) or DMSO for 8 hours and % mitotic cells was measured by flow cytometry. Mitotic cells were detected with antibodies against MPM2 (20,000 cells per cell line, black lines are median values, Mann-Whitney test p=0.54). **f,** Cohesion between sister chromatids was assessed by measuring inter-chromatid distance at the centromere from metaphase spreads (shown in inset images), n=25 sister chromatid pairs across 5 metaphases. HCT-116 cells were transfected with siRNA targeting the cohesin subunit SCC1 as a positive control. **g,** % Prometaphase cells with >4 centrioles, detected with anti-centrin 3 antibodies (n=100 cells per cell line, black lines are median values, statistic: Mann-Whitney test p=0.59). **h,** images of HT55 and SKCO1 cells stained with anti-β–tubulin and anti-centrin 3 antibodies, displaying a multipolar and bipolar spindle respectively. **i,** % Prometaphase cells with multipolar spindles, detected with antibodies against β-tubulin (n=100 cells per cell line, black lines are median values, statistic: Mann-Whitney test p=0.027). **j,** % Metaphases in cell lines as indicated displaying a chromosome of the indicated morphology, measured from chromosome spreads hybridised to all-centromere DNA probe.



Supplementary Figure 3. Pharmacological induction of replication stress induces structural and numerical instability. a-m, HCT-116 cells were treated with 0.2 μM aphidicolin for 24 h. **a,** % Anaphases with segregation errors (mean±s.e.m, 3 experiments, n>50 anaphases per experiment. **b,** Segregation error classification. **c,** Examples of an acentric chromosome and an anaphase bridge induced by aphidicolin treatment. **d,** % Metaphases with structurally abnormal chromosomes. **e,** Classification of structurally abnormal chromosomes. **f,** Example of an aphidicolin-induced chromosome break. **g,h,** % Deviation from the modal copy number for centromeres 2 and 15 across the HCT-116 cell population with representative image for aphidicolin treatment shown **(h)**. **i,** % Prometaphases with ≥3 γH2AX foci (mean±s.e.m of 3 independent experiments, n=100 cells). **j,** Representative image of prometaphase DNA damage after aphidicolin treatment (cells stained with antibodies against γH2AX). **k,** UFBs in an anaphase cell after aphidicolin treatment (cells stained with antibodies against RPA). **l,** % Anaphases with UFBs ± aphidicolin (black bars: anaphases without errors, blue bars: anaphases with errors) (mean±s.e.m of three independent experiments, n=100 cells per experiment). **m,** % G1 cells with ≥ 3 53BP1 bodies (mean±s.e.m of three independent experiments, n>150 cells per experiment). All statistical tests were two-tailed t-tests **p<0.01, ***p<0.001.



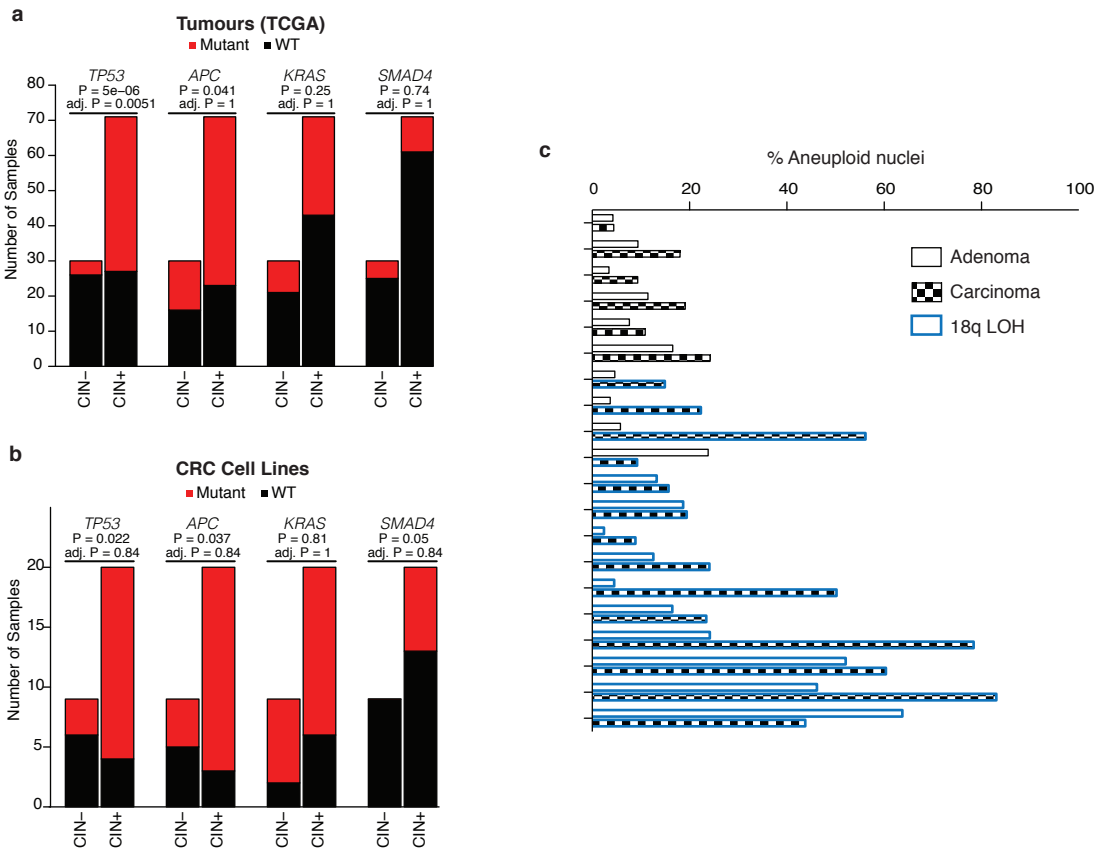
Supplementary Figure 4. Prometaphase DNA damage in CIN+ cells is not solely attributable to telomeric or oxidative stress-induced DNA damage. a,b, Levels of the DNA modification 8-oxoguanine, a marker of oxidative DNA damage, were measured by flow cytometry in a panel of CIN+ and CIN- cell lines. **a,** Median CV values for 8-oxoguanine staining (values are the mean of two independent experiments). Treatment with hydrogen peroxide (H_2O_2) (350 μM for 4 h, followed by a 14 h recovery period) to induce oxidative stress, verified the specificity of the staining - representative experiment shown. **(b)**. **c,** 8-oxoguanine levels were assayed by immunohistochemistry. Representative fields of cells (40x magnification) are shown (± H_2O_2 treatment as in **(b)**); no differences were observed between CIN+ and CIN- cell lines. **d,** Chromosome spreads were prepared from four CIN+ cell lines and stained with antibodies for γ H2AX and centromeres (ACA). **right**: Foci were classified as **i)** interstitial foci (clearly not at the chromosome end) or **ii)** terminal foci (at the end of the chromosome arm). Short chromosome arms **(iii)** could not be accurately scored and were therefore excluded from this classification. **e,** Quantification of percentage of foci that were interstitial or terminal (n >200 foci per cell line).



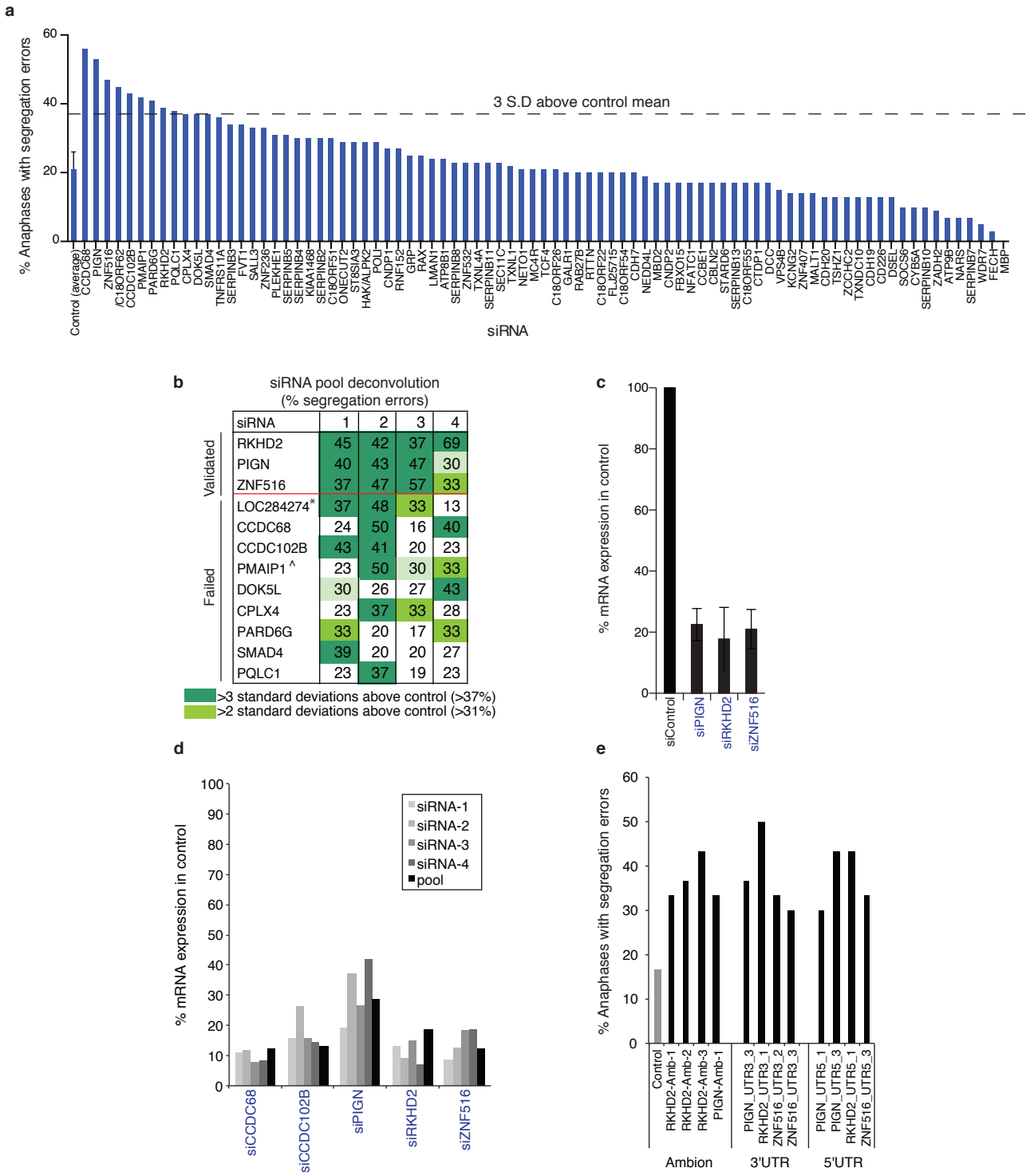
e

	Cell line	1st label origins	2nd label origins	Progressing forks	2nd label Terminations	1st label terminations/ Stalled forks	Fork assymmetry (sister fork ratio <0.75)
CIN-	RKO	22.8	1.2	60.8	14.6	0.6	29.1
	HCT-116	21.9	5.0	57.1	13.7	2.3	24.1
CIN+	SW620	18.8	1.8	64.5	10.7	4.2	44.4
	HT55	18.9	1.5	63.9	12.0	3.7	25.4
	HT29	20.9	1.6	60.5	15.3	1.7	35.1
	SW1116	21.9	2.8	56.2	16.4	2.8	34.1

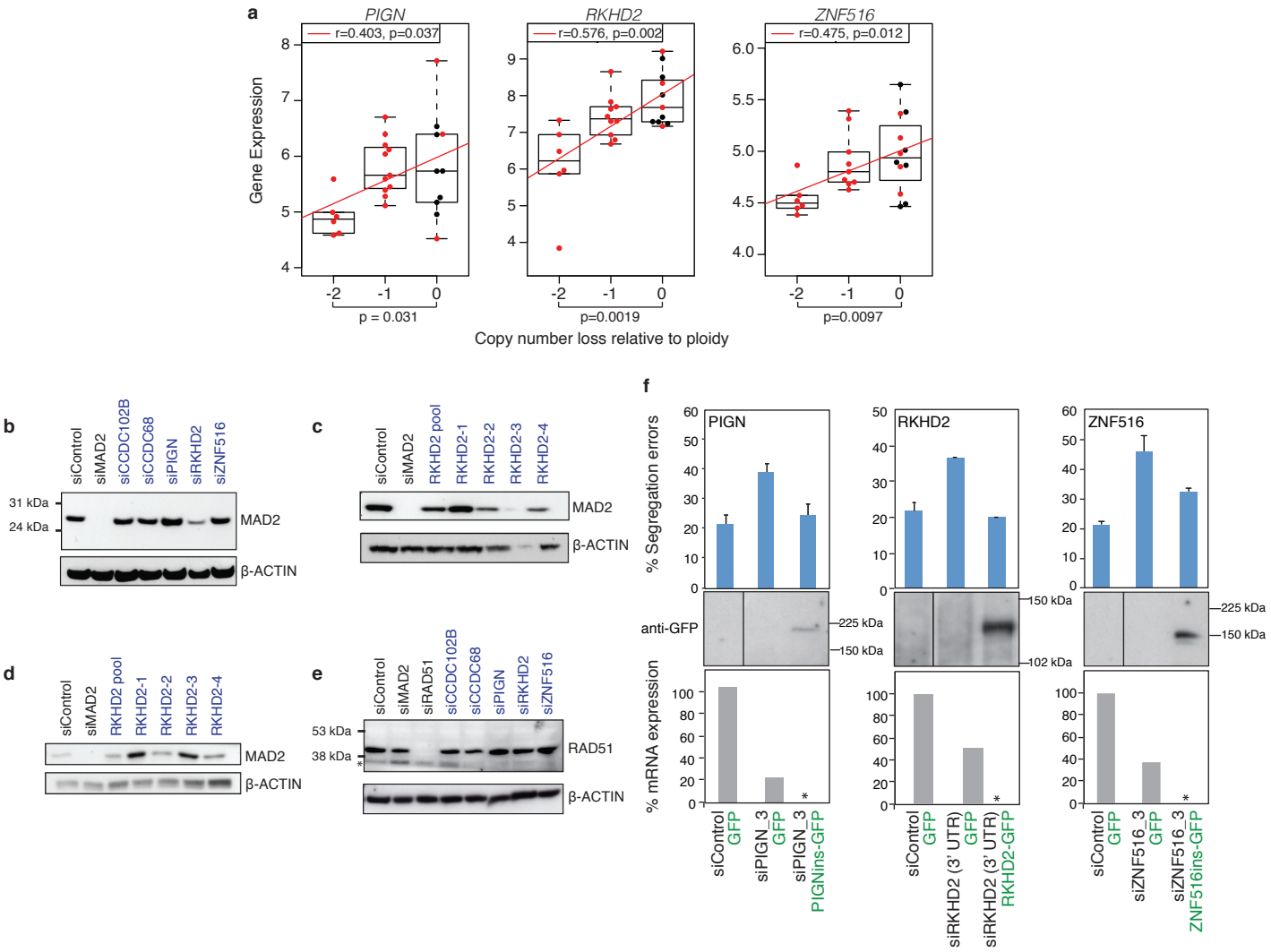
Supplementary Figure 5. CIN+ cells exhibit reduced DNA replication fork rates and evidence of fork stalling. a-e, CIN+ and CIN- cells were pulsed with CldU followed by IdU (30 minutes each), before DNA fibre assays were performed: **a,** Mean replication fork rate for both CldU and IdU incorporation (n>60 forks per experiment, mean±s.e.m of three independent experiments). **b,** % Replication forks exhibiting speeds ≤0.8 kb/min (n>60 forks per experiment, mean±s.e.m of three independent experiments). **c,** Schematic of replication structures. **d,** Representative fibres showing asymmetric sister replication fork progression. Sister fork ratio = x/y (where x is the shorter fork). Forks with a ratio of <0.75 were considered asymmetric[5]. **e,** Table showing quantification of replication structures and sister fork asymmetry in CIN+ and CIN- cell lines (n=600 replication forks per cell line or n=51-142 bidirectional forks per cell line for fork asymmetry, from three independent experiments). No increase was observed in the frequency of origin structures (1st label origins, 2nd label origins) despite reports that slow replication fork rates are accompanied by increased origin firing, in order to enable the timely completion of DNA replication[5].



Supplementary Figure 6. TP53 is the only gene significantly correlated with CIN. a, Exome sequencing data was analysed for the TCGA cohort of colorectal tumours (n=101)[19]. Tumours were defined as CIN+ or CIN- based on weighted genome instability index >0.2 (see Methods) and genes that showed significant enrichment (Fisher's exact test) for mutation in CIN+ tumours were identified. Only *TP53* remained significant after correction for multiple testing[34], and *TP53* mutations also occurred in CIN- tumours. **b,** Data examining the mutation status of 64 protein-coding genes in each of 29 CRC cell lines was downloaded from COSMIC (http://cancer.sanger.ac.uk/cancergenome/projects/cosmic/) and the relationship of mutation status to CIN status examined. Three genes were more frequently mutated in CIN+ versus CIN- cell lines: *TP53, SMAD4* and *APC*, although none remained significant after correcting for multiple testing. *TP53* and *APC* mutations both occurred in CIN- cell lines. **c,** % Aneuploid nuclei in preparations of nuclei from paired, adjacent colorectal adenomas and carcinomas. Nuclei were stained with Feulgen stain, and DNA content was measured by DNA image cytometry. 18q LOH was determined by microsatellite or SNP analysis (see Methods).

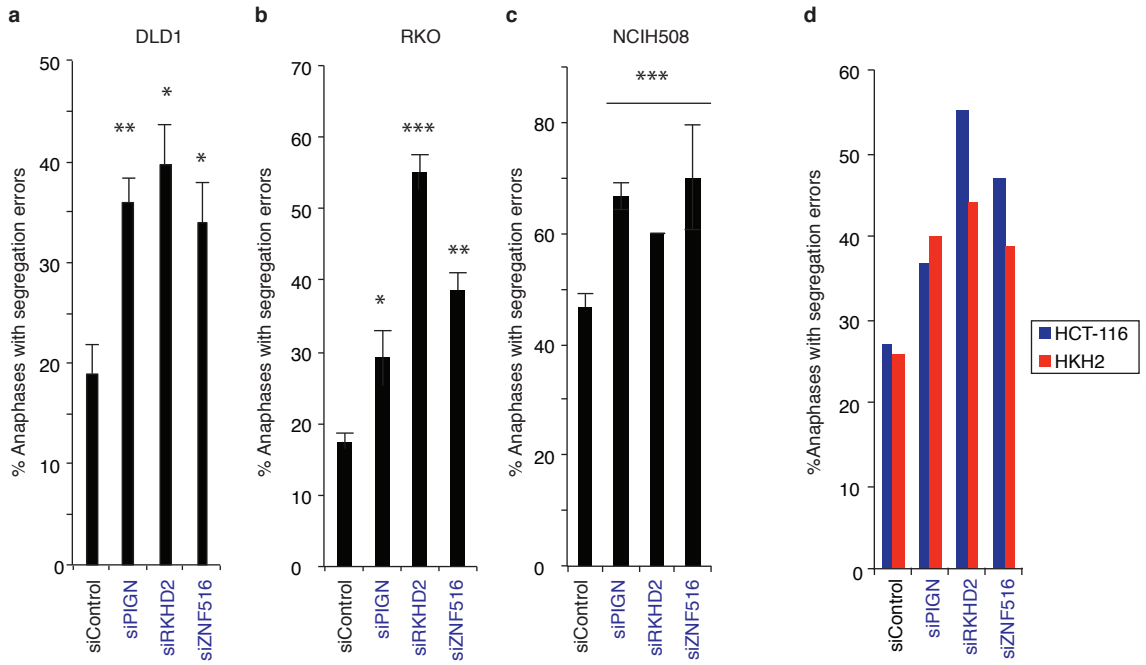


Supplementary Figure 7. Identification and validation of CIN-suppressor genes. a, % Anaphase cells with segregation errors in HCT-116 cells following silencing of 94 genes encoded on 18q (n>30 anaphases per siRNA pool). 8 genes encoding hypothetical proteins (for which siRNA pools were unavailable) were not incuded and 3 genes could not be assessed for chromosome segregation errors (see Supplementary Table 3). **b,** siRNA pool deconvolution: segregation error rates after transfection with the four individual siRNAs from the pool used in **(a).** *LOC284274 mRNA could not be detected and thus was excluded as it was not possible to validate silencing or determine whether this gene is expressed. ^PMAIP1 was excluded due to previously reported off-target effects upon MAD2 protein of sequences 2 and 3[40]. **c,** mRNA expression measured by quantitative reverse-transcription PCR relative to expression in control-transfected HCT-116 cells for each siRNA pool validating with 4/4 sequences (mean±s.d of 3 experiments). **d,** mRNA expression relative to control-transfected cells after transfection of individual sequences comprising the siRNA pools for which at least two sequences induced segregation errors beyond the screen threshold (37%). CCDC68 and CCDC102B were excluded from further functional follow-up due to discrepancies between mRNA knockdown and segregation error phenotype. **e,** % Segregation error frequency upon silencing the indicated gene with further independent siRNA sequences targeting the coding sequence, and with sequences targeting the 3' and 5'-untranslated regions of the transcripts (n>30 anaphases per sequence).

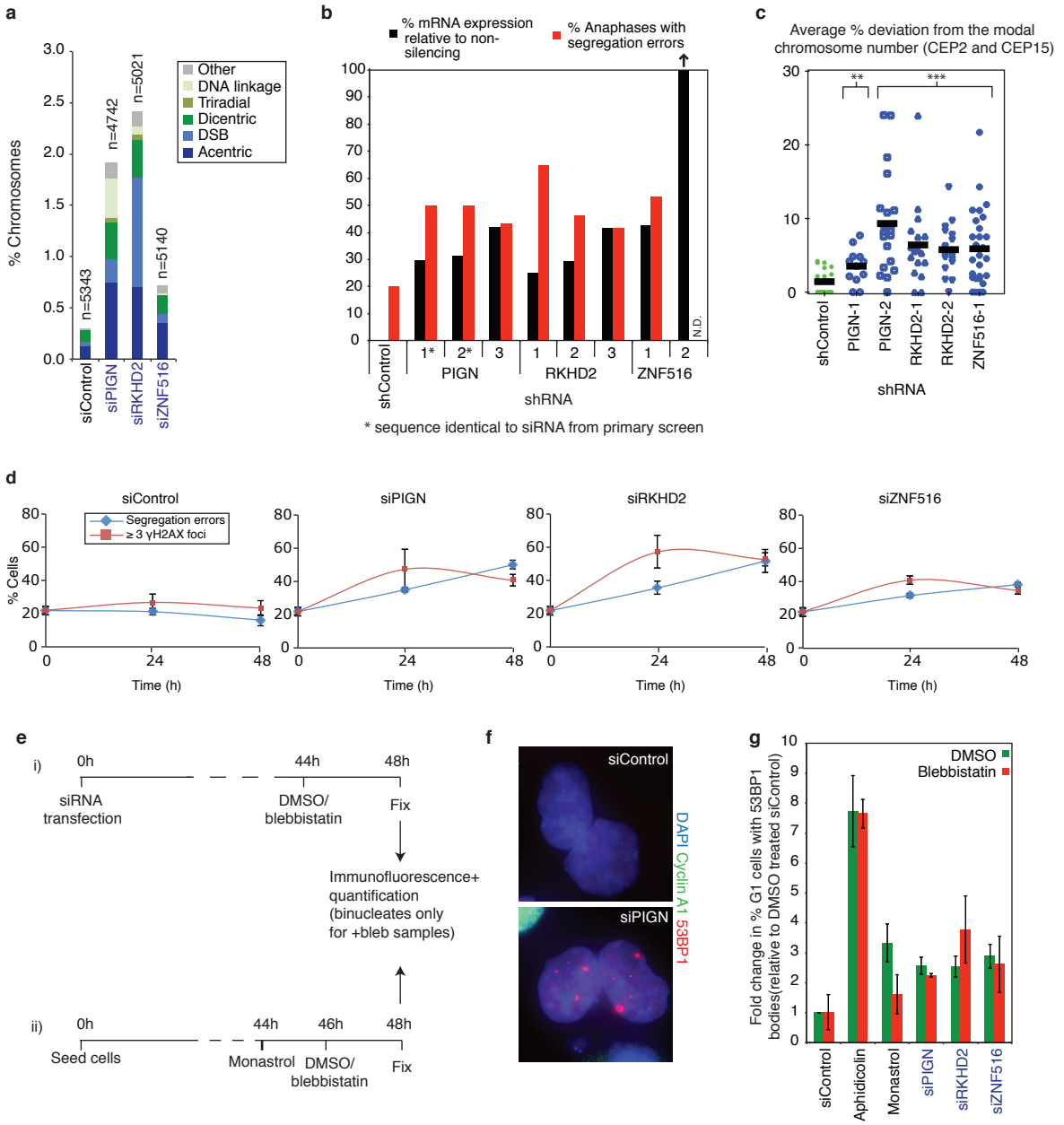


Supplementary Figure 8. CIN-suppressor gene validation. a, Spearman's rank correlation between mRNA expression and DNA copy number for *PIGN, RKHD2* and *ZNF516* in CIN- (n=8, black dots) and CIN+ (n=20, red dots) cell lines. Statistic: Wilcoxon Mann-Whitney test. **b,** MAD2 protein levels in HCT-116 cells transfected with siRNA pools with at least 2 sequences inducing ≥ 37% anaphases with segregation errors. **c,** MAD2 protein levels assessed after transfection of the individual RKHD2 sequences. Sequences 2 and 4 exhibited partial depletion of MAD2 and were therefore excluded. Lane 6 (sequence 3) was underloaded and therefore samples were rerun **(d)** to verify MAD2 levels were not depleted upon transfection with this sequence. **e,** RAD51 protein levels following validated siRNA transfection (*denotes non-specific band). **f,** HCT-116 cells were transfected with siRNAs as indicated and RKHD2-GFP or siRNA-insensitive versions of PIGN-GFP (PIGNins-GFP) or ZNF516-GFP (ZNF516ins-GFP) before scoring segregation errors (n>30 anaphases per experiment). Mean+s.e.m of three independent experiments (PIGN and ZNF516) or mean+S.D. of two independent experiments (RKHD2) is shown (**top panels**). Western blotting using anti-GFP confirmed the expression of the GFP-tagged proteins (**middle panels**) and RT-qPCR confirmed gene knockdown (**lower panels**). Western blot and qPCR data are from the same experiment. *Endogenous gene knockdown could not be assessed in these samples due to the confounding presence of exogenous siRNA-insensitive mRNA transcripts. NB: PIGN-GFP and RKHD2-GFP observed MW are larger than predicted MW, in agreement with previous reports (RKHD2[41]), and post-translational modification (PIGN, data not shown).

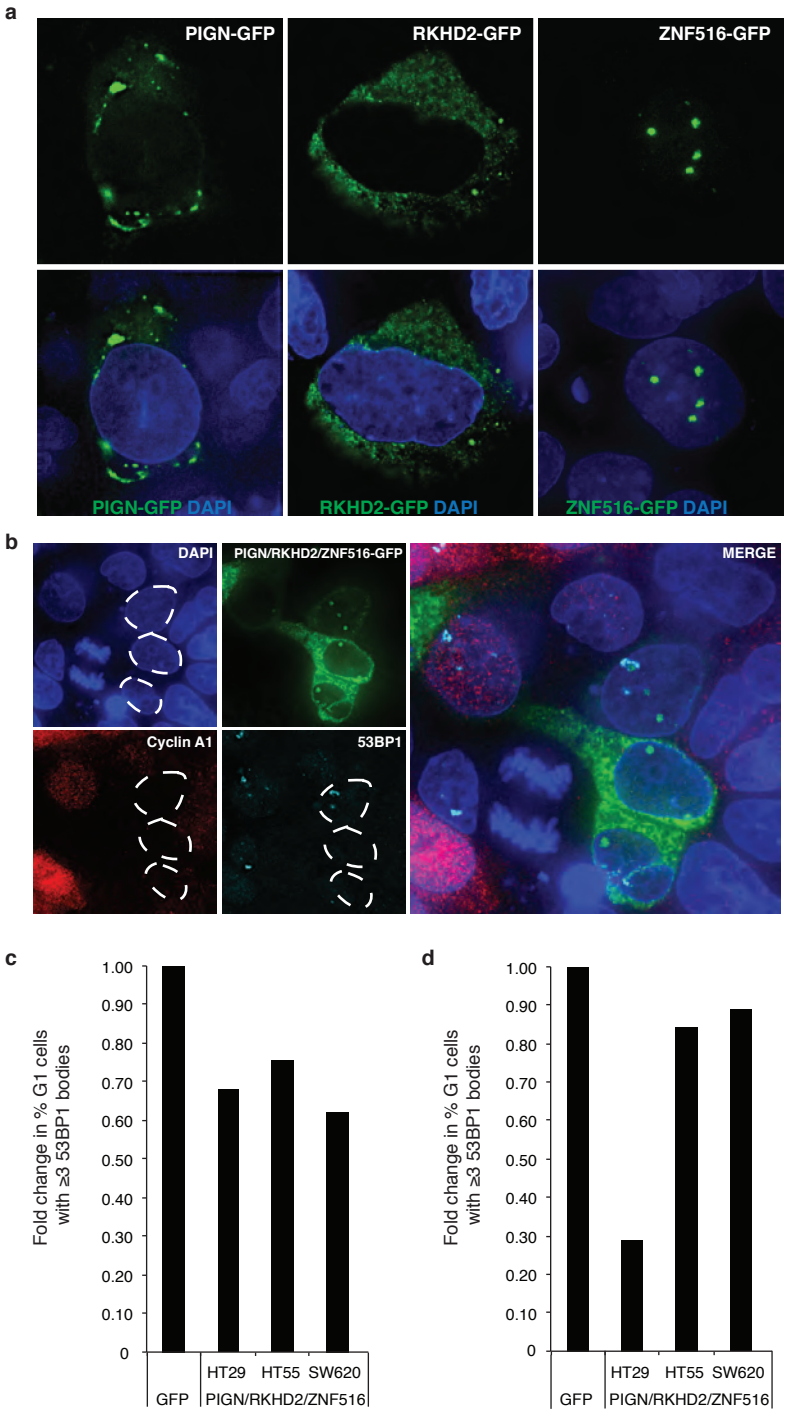
Methods: PIGN-GFP and ZNF516-GFP were extracted using the Qproteome cell compartment kit (Qiagen).



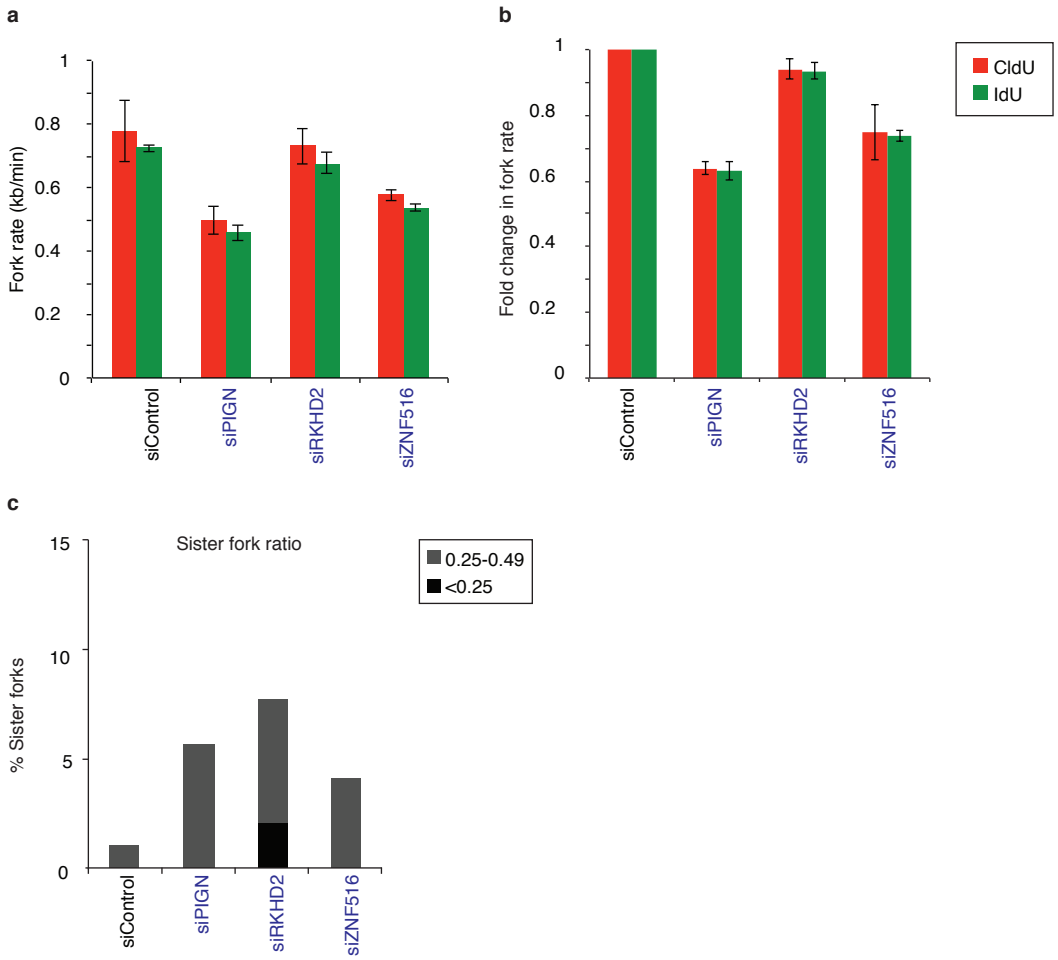
Supplementary Figure 9. CIN-suppressor gene-silencing induces segregation errors in additional cell lines. a-c, Segregation error frequencies in **(a)** DLD1 (CIN-) **(b)** RKO (CIN-) and **(c)** NCIH508 (CIN+, 18q normal) cells following transfection with validated siRNA pools (mean±s.e.m of 3 experiments, n>30 anaphases per condition per experiment, two-tailed t-test,*p<0.05, **p<0.01,***p<0.001). **d,** Segregation error frequency in HCT-116 (KRAS mutant) and isogenic cell line HKH2 (which carries a wild-type KRAS allele), n>60 anaphases per condition, two experiments.



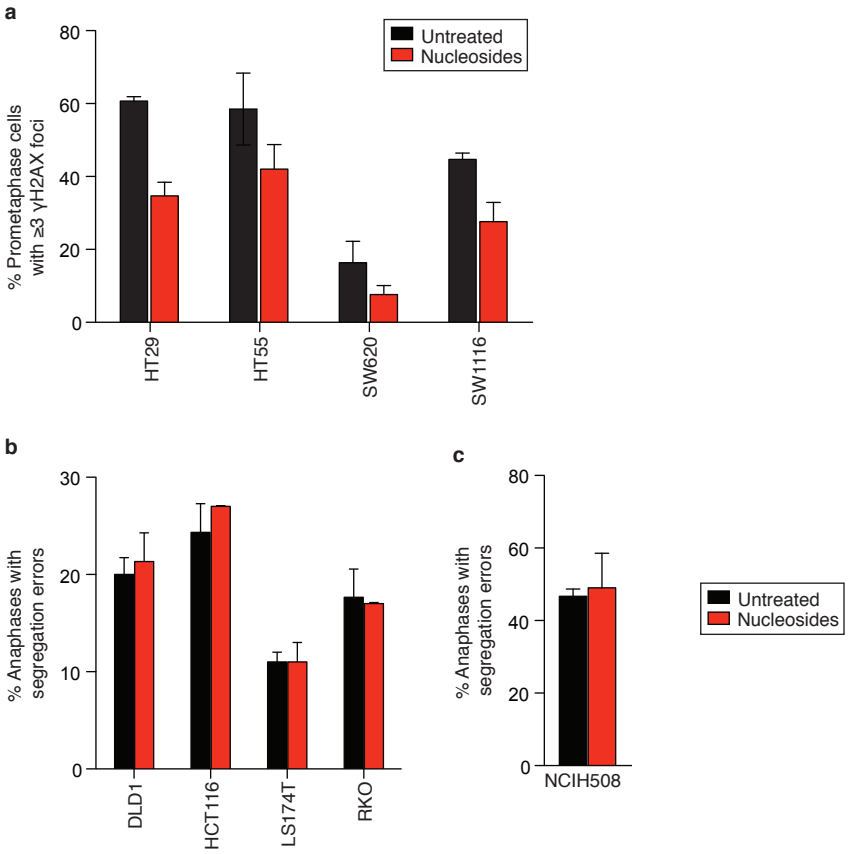
Supplementary Figure 10. CIN-suppressor gene silencing induces structural and numerical instability. a, % Aberrant chromosomes identified on metaphase chromosome spreads following CIN-suppressor silencing, categorised by type of abnormality. **b,** mRNA expression and anaphase segregation error frequencies (n=50 anaphases) following shRNA mediated silencing of PIGN, RKHD2 and ZNF516 (N.D. not determined). **c,** HCT-116 cell lines stably expressing shRNAs against the three CIN-suppressor genes were seeded at low density on glass slides to allow colony formation. Slides were then hybridised to fluorescently labelled centromeric probes to chromosomes 2 (CEP2) and 15 (CEP15) and stained with DAPI. Each point is the mean % deviation from the modal centromere copy number for the two centromeric probes for an individual colony. Lines are median values, statistical test: Mann-Whitney test, **p<0.01, ***p<0.001. **d,** HCT-116 cells were transfected with siRNAs as indicated and fixed and stained with antibodies to γH2AX and β-tubulin at 24 and 48 h post siRNA transfection. Segregation errors and prometaphase DNA damage were scored (mean±s.e.m, n=50 prometaphases, 30 anaphases per experiment) of four (siControl) or three independent experiments (siPIGN, siRKHD2 and siZNF516), and compared to untransfected cells (t=0) (mean±s.d. of two independent experiments). Both phenotypes occur with similar kinetics, making it unlikely that lagging chromosomes undergoing cytokinesis-induced DNA damage explain the induction of prometaphase DNA damage following CIN-suppressor silencing. **e,** Schematic of experiment for testing whether 53BP1 bodies observed in G1 cells were attributable to DNA damage during cytokinesis. siRNA-depleted cells were treated with blebbistatin (100 μM) or DMSO for 4 h, 44 h post-transfection, to inhibit cytokinesis. As a control, cells were treated with 0.2 μM aphidicolin for 24 h prior to blebbistatin addition (aphidicolin-induced 53BP1 bodies should be unaffected by cytokinesis inhibition[15]). In addition, cells were treated with monastrol (100 μM, 2 h) to induce lagging chromosomes, then released into blebbistatin or DMSO for 2 h, as a control for 53BP1 foci that are reduced by cytokinesis inhibition[7]. Cells were then fixed and stained, and only binucleate cells scored to ensure that cells had passed through mitosis in the presence of a cytokinesis block. Blebbistatin treatment did not affect mitotic entry (data not shown). **f,** Example images of binucleate control and PIGN-depleted cells after blebbistatin treatment. **g,** Quantification of two independent experiments (n>100 binucleate G1 cells (blebbistatin), or 200 G1 cells (DMSO), per condition), normalised to DMSO-treated control cells. Only G1 53BP1 foci induced by monastrol were reduced by cytokinesis inhibition.



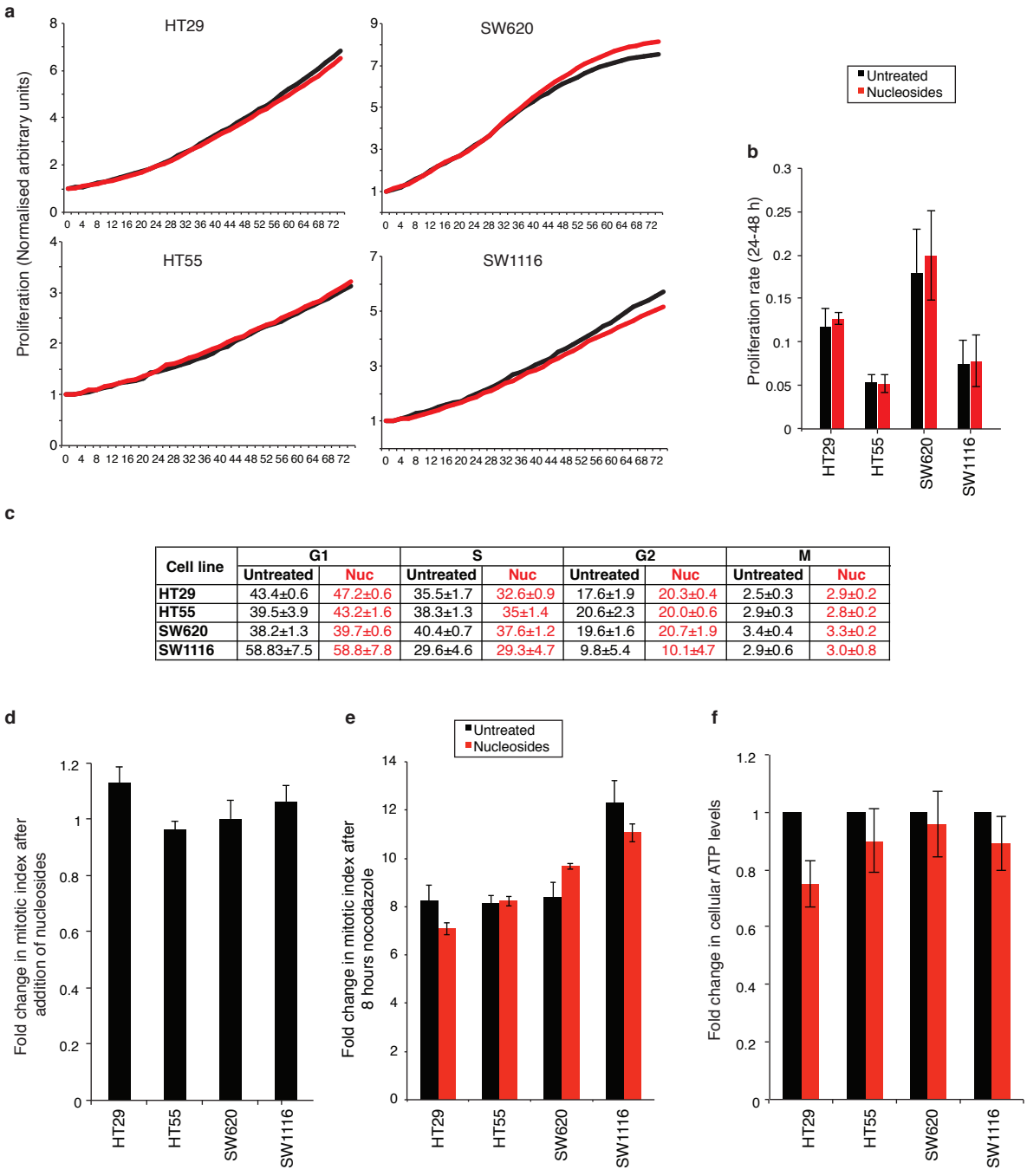
Supplementary Figure 11. 53BP1 bodies in G1 cells are partially reduced upon exogenous expression of CIN-suppressor genes in CIN+ cells with 18q loss. a, Example images of GFP-tagged CIN-suppressors as indicated in HT55 cells. Localisation patterns are consistent with published reports for PIGN and RKHD2[41-43]. **b,** Representative image from CIN+ HT55 cells co-transfected with GFP-tagged PIGN, RKHD2 and ZNF516. **c,d,** Quantifications of G1 53BP1 bodies in PIGN/RKHD2/ZNF516-GFP transfected cells, relative to GFP-transfected cells. Only GFP positive G1 (cyclin-A1 negative) cells were scored (n>100 cells per experiment).



Supplementary Figure 12. CIN-suppressor silencing impairs DNA replication fork progression in HCT-116 cells. a-c, 48 h after transfection with siRNA pools, cells were sequentially pulsed for 30 mins each with CldU and IdU, before DNA fibre assays were performed. **a,** Mean replication fork rate for CldU and IdU (mean±s.d. of two independent experiments, n>70 forks per experiment (total of >200 forks assessed)). **b,** Mean fork rate normalised to control replication fork rate (mean±s.d. of two experiments, n>70 forks per experiment, total of >200 forks assessed). **c,** Quantification of sister fork asymmetry following CIN-suppressor silencing (sister fork ratio= x/y where x is the shorter sister fork, n=68 -141 bidirectional forks per siRNA). Only the most asymmetric forks are shown (ratio <0.49).



Supplementary Figure 13. Nucleoside supplementation of CIN+ cells reduces prometaphase DNA damage and chromosome segregation errors, but does not reduce segregation error frequency in CIN- cells. a, % Cells with ≥3 γH2AX foci ± nucleoside supplementation for 48 h (mean+s.e.m of three (HT29, SW620, SW1116) or four (HT55) independent experiments, n=100 cells per experiment). **b,** % Anaphases with segregation errors ± nucleoside supplementation for 48 h in CIN- cell lines as indicated (mean+s.e.m of three independent experiments, n≥30 anaphases per cell line per experiment). **c,** % Anaphases with segregation errors ± nucleoside supplementation for 48 h in NCIH508 CIN+, 18q normal cells, (mean+s.e.m of three independent experiments, n≥30 anaphases per experiment).



c

Cell line	G1		S		G2		M	
	Untreated	Nuc	Untreated	Nuc	Untreated	Nuc	Untreated	Nuc
HT29	43.4±0.6	47.2±0.6	35.5±1.7	32.6±0.9	17.6±1.9	20.3±0.4	2.5±0.3	2.9±0.2
HT55	39.5±3.9	43.2±1.6	38.3±1.3	35±1.4	20.6±2.3	20.0±0.6	2.9±0.3	2.8±0.2
SW620	38.2±1.3	39.7±0.6	40.4±0.7	37.6±1.2	19.6±1.6	20.7±1.9	3.4±0.4	3.3±0.2
SW1116	58.83±7.5	58.8±7.8	29.6±4.6	29.3±4.7	9.8±5.4	10.1±4.7	2.9±0.6	3.0±0.8

Supplementary Figure 14. Nucleoside supplementation does not affect proliferation, cell cycle distribution or cellular ATP levels. a, CIN+ cell lines as indicated were supplemented with nucleosides and then imaged in 96 well plates every 2 h using an in-situ imaging system. The % confluency of the well was calculated at each timepoint, and growth curves constructed. Representative growth curves for each cell line are shown. **b,** Mean proliferation rates ± nucleosides (mean±s.e.m of three independent experiments). Proliferation rates were calculated by measuring the gradient of growth curves between 24 and 48 h. **c,** Table showing cell cycle distribution ± nucleoside supplementation for 48 h (mean±s.e.m of three independent experiments). **d,** Fold change in mitotic index ± nucleoside supplementation for 48 h, measured by flow cytometry (mean±s.e.m of three independent experiments). **e,** Fold change in mitotic index upon nocodazole treatment (8 h, 100 ng/ml) relative to DMSO treatment ± nucleoside supplementation for 48 h (mean±s.e.m of three independent experiments). **f,** ATP levels were measured using Cell Titer Glo (CTG). CTG values were then normalised to cell biomass (sulforhodamine B solution). Fold-change in ATP levels was calculated for nucleoside-treated cells by normalising the values to those of untreated cells (mean±s.e.m of three experiments).

Supplementary Table 1: Frequency of 18q loss in CIN+ CRC cell lines and aneuploid colorectal tumours (BAC array and TCGA cohort[19]).

	CRC Cell Lines (CIN+, SNP 6.0) n = 20	Aneuploid Tumours (BAC Array) n = 26	TCGA Tumours (CIN+, SNP.6.0) n = 103
18q loss (>50% of 18q)	80%	88%	82%
18q loss (>75% of 18q)	70%	85%	80%

Supplementary Table 2: Genes encoded in regions of copy number loss in CIN+ CRC (Excel datasheet).
List of 1331 genes encoded in regions of significant copy number loss in both aneuploid tumours and CIN+ CRC cell lines ($p<0.001$, Fisher's exact test).

Supplementary Table 3: Genes included in RNA interference screen (Excel datasheet).
List of 94 genes on 18q that were present at one or no copies in ≥30% of CIN+ cell lines, and no more than one CIN- cell line, which were included in the RNA interference screen.

Supplementary Table 4: Percentage of cell lines and tumours that have lost zero, one, two or all three CIN-suppressor genes.

Number of CIN-suppressors lost relative to median ploidy	CRC Cell Lines (CIN+, SNP 6.0) n = 20	TCGA Tumours (CIN+, SNP.6.0) n = 103
Zero	15%	16%
One	0%	1%
Two	15%	4%
All three	70%	79%

Supplementary Table 5 (Excel datasheet): Correlation of mRNA expression and copy number in tumours (TCGA cohort). Spearman's rank correlation co-efficient for the 75 genes that were assessable on gene expression microarrays of the 94 genes on 18q included in the RNA interference screen.

Supplementary Table 6: SiRNA sequences used in this study (sense sequence).

PIGN Dharmacon D-012463-01	GAGACCACGUUUAUACAUA
PIGN Dharmacon D-012463-02	AUUGACUCAUCAGUUGACU
PIGN Dharmacon D-012463-03	UUAAGCAUUUCACGUAGAC
PIGN Dharmacon D-012463-04	AUUAUGAAGUUAAGGGUCC
RKHD2 Dharmacon D-006989-01	UUUCGUCCCUAUAUAUAAC
RKHD2 Dharmacon D-006989-02	UUGAUCCCGUCGUUAUUGG
RKHD2 Dharmacon D-006989-04	AUUGGGAGGCUCGUUCACC

RKHD2 Dharmacon D-006989-17	AUAUGCACAAACAGAACCG
ZNF516 Dharmacon D-010660-01	AAACAUUUGUCCAAGGGCG
ZNF516 Dharmacon D-010660-02	AUUGGUAACCCGACUUAGG
ZNF516 Dharmacon D-010660-03	UUGCUGAUUUCGGCGGACG
ZNF516 Dharmacon D-010660-04	AAAGGCUCCGCAAUAGAGG
PIGN 3' UTR oligo 3	GCUGAUGCUAAGCUUUCCU
PIGN 5' UTR oligo 1	GCUCCAGAAACAGUAUUAU
PIGN 5' UTR oligo 3	CCGUUUGAUCUUGGCAAUU
RKHD2 3' UTR oligo 1	GCAUUACAGACGAUUCAUA
RKHD2 5' UTR oligo 1	GCUGGUUAAGGCCAUGAAA
ZNF516 3' UTR oligo 2	CCAUUCGAACUGAGCGAAU
ZNF516 3' UTR oligo 3	CCAUGCAGUGUACGCAAUU
ZNF516 5' UTR oligo 3	GCUCUUCUGACCAUGGAGA
Ambion PIGN S24088	CCUUUACCCUGGGAAUUGA
Ambion RKHD2_1 S27947	GCCUGAAAAUGUUGACCGA
Ambion RKHD2_2 S27948	GGAUUAUCAUGUAGUCCUA
Ambion RKHD2_3 S27949	GGAGUGAUCCUUCUGGUAA

Supplementary Table 7: shRNA sequences used in this study.

PIGN – 1 (V2LHS_244784) pGIPZ	TATGTATAAACGTGGTCTC
PIGN – 2 (V2LHS_246658) pGIPZ	AATTCGTAAAGTGCATCTG
PIGN – 3 (V2LHS_50417) pGIPZ	TCATCTAATTCGTAAAGTG
RKHD2 – 1 (V2LHS_235038) pGIPZ	ATGCATTTCTATTTCTTCC
RKHD2 – 2 (V2LHS_135328) pGIPZ	AATTGGATATCATTCTTGC
RKHD2 – 3 (V2LHS_234137) pGIPZ	TTGGATATCATTCTTGCGC
ZNF516 – 1 (V2HS_7966) pSM2c	TTAAGTAAACTGTTGCTCTCGC
ZNF516 – 2 (V2HS_7968) pSM2c	TATTCTTAGGCATGCTAGCGG
ZNF516 – 3 (V2HS_79671) pSM2c	TAACAAGGAGAGGCATCTGCC

Supplementary References

40. Tsui, M. *et al.* An intermittent live cell imaging screen for siRNA enhancers and suppressors of a kinesin-5 inhibitor. *PLoS One* **4**, e7339, doi:10.1371/journal.pone.0007339 (2009).
41. Buchet-Poyau, K. *et al.* Identification and characterization of human Mex-3 proteins, a novel family of evolutionarily conserved RNA-binding proteins differentially localized to processing bodies. *Nucleic Acids Res* **35**, 1289-1300, doi:gkm016 [pii]
10.1093/nar/gkm016 (2007).
42. Hong, Y. *et al.* Pig-n, a mammalian homologue of yeast Mcd4p, is involved in transferring phosphoethanolamine to the first mannose of the glycosylphosphatidylinositol. *J Biol Chem* **274**, 35099-35106 (1999).
43. Lee, M. G., Wynder, C., Cooch, N. & Shiekhattar, R. An essential role for CoREST in nucleosomal histone 3 lysine 4 demethylation. *Nature* **437**, 432-435, doi:nature04021 [pii]
10.1038/nature04021 (2005).

Sweave Document: Replication stress links structural and numerical cancer chromosomal instability

Burrell & McClelland et al.

January 15, 2013

The following Sweave document describes the data analysis.
All data analysis was performed in the R statistical environment.

1 Getting started

1.1 Loading required packages

```
> rm(list = ls())
> library(limma)
> library(qvalue)
> library(siggenes)
> library(gregmisc)
> library(beeswarm)
> library(fields)
> library(multtest)
> library(hthgu133a.db)
```

1.2 Required Functions

```
> ################################################
> # gii score weighted by chromosome length
> ################################################
>
> gii.weight.chrom <- function(x, seg, ploidy){
+         gii.weight <- 0
+         ploidy.x <- ploidy[x]
+         sub <- subset(seg, seg[, 1] == x)
+
+         for(cc in unique(seg[, 2])){
+          sub.chrom <- subset(sub, sub[, 2] == cc)
+          a <- sub.chrom[as.numeric(sub.chrom[, 6]) > ploidy.x
```

```
+                         | as.numeric(sub.chrom[, 6]) < ploidy.x, ]
+      b <- sum(as.numeric(sub.chrom[, 4]) - as.numeric(sub.chrom[, 3]))
+      if(!is.null(dim(a)))
+                  gii.chrom <- sum(as.numeric(a[, 4]) - as.numeric(a[, 3]))/b
+          else
+                  gii.chrom <-  sum(as.numeric(a[4]) - as.numeric(a[3]))/b
+
+          gii.weight <- gii.weight + gii.chrom
+     }
+         return(gii.weight/length(unique(seg[, 2])))
+ }
> ###################################
>
> weighted.quantile <- function(x, w, probs = seq(0, 1, 0.25)
+                                 , na.rm = FALSE, names = TRUE) {
+
+     if (length(x) != length(w))
+       stop("Weights and variable vectors are of unequal length!")
+     if (na.rm)
+       { valid.obs <- !is.na(x) & !is.na(w)
+         x <- x[valid.obs]
+         w <- w[valid.obs]}
+     else if (any(is.na(x)) | any(is.na(w)))
+       stop("Missing values and NaN's not allowed if `na.rm' is FALSE")
+     if (any((p.ok <- !is.na(probs)) & (probs < 0 | probs > 1)))
+       stop("probs outside [0,1]")
+     if (na.p <- any(!p.ok)) {
+       o.pr <- probs
+       probs <- probs[p.ok]
+     }
+     np <- length(probs)
+     if (missing(w))
+       w <- rep(1, length(x))
+     if (na.rm) {
+       w <- w[i <- !is.na(x)]
+       x <- x[i]
+     }
+     # check if weights are probs or unit correspondences
+     ind <- order(x)
+     cumw <- cumsum(w[ind])/sum(w)
+     x <- x[ind]
+     med <- numeric(0)
+     for(i in 1:length(probs)) {
+       if(probs[i] > 0 & probs[i] < 1) med[i] <- max(x[cumw<=probs[i]])
+       else if (probs[i] == 0) med[i] <- min(x)
+       else if (probs[i] == 1) med[i] <- max(x)
```

```
+     }
+     # the treatment of names
+     med
+   }
> ###############################################################
> weighted.median <- function(x,w, na.rm = FALSE) {
+   if (missing(w))
+     w <- rep(1, length(x))
+   if (na.rm) {
+     w <- w[i <- !is.na(x)]
+     x <- x[i]
+     }
+   weighted.quantile(x,w,.5)
+ }
> ###############################################################
>
> function.matend = function(dataend){
+   start = 1
+   end = c()
+   for(i in 2:nrow(dataend)){
+     if(
+       length(which(dataend[i,c(1, 3:ncol(dataend))]
+                         !=  dataend[i - 1,c(1, 3:ncol(dataend))])) > 0
+       ){
+       end = c(end, i-1)
+       start = c(start, i)
+      }
+   }
+   end = c(end, i)
+   startend = cbind(start, end)
+   matend = matrix(0, nr = 0, nc = ncol(dataend) + 1)
+   for(i in 1:nrow(startend)){
+     if(startend[i, 1] != startend[i, 2])
+       matend = rbind(matend, c(mean(dataend[startend[i, 1]:startend[i, 2], 1])
+       , dataend[startend[i, 1], 2], dataend[startend[i, 2], 2]
+       , apply(dataend[startend[i, 1]:startend[i, 2], 3:ncol(dataend)], 2, mean)))
+     else{
+       matend = rbind(matend, c(mean(dataend[startend[i, 1]:startend[i, 2], 1])
+       , dataend[startend[i, 1], 2], dataend[startend[i, 2], 2]
+       , dataend[startend[i, 1]:startend[i, 2], 3:ncol(dataend)])  )
+     }
+   }
+
+
+ return(matend)
+ }
```

```
> ########################
>
> fun.find.prob.values <- function(name, seg, ploidy, gain = TRUE){
+         sub.seg <- subset(seg, seg[, 1] == name)
+         tmp.copy <- as.numeric(sub.seg[, 6]) - ploidy[name]
+         tmp.probes <- as.numeric(sub.seg[, 5])
+         if(gain){
+                 un.copy.gain <- sort(unique(tmp.copy[tmp.copy > 0]))
+                 mat.copy.gain <- matrix(NA, nr = length(un.copy.gain), nc = 2)
+                 for(k in 1:length(un.copy.gain))
+                         mat.copy.gain[k, ] <- c(un.copy.gain[k]
+                                 , sum(tmp.probes[tmp.copy == un.copy.gain[k]]))
+                 mat.copy.gain <- rbind(c(0, sum(tmp.probes[tmp.copy <= 0]))
+                                         , mat.copy.gain)
+                 mat.copy.gain[, 2] <- mat.copy.gain[, 2]/sum(tmp.probes)
+                 return(mat.copy.gain)
+         }else{
+                 un.copy.loss <- sort(unique(tmp.copy[tmp.copy < 0]))
+                 mat.copy.loss <- matrix(NA, nr = length(un.copy.loss), nc = 2)
+                 for(k in 1:length(un.copy.loss))
+                         mat.copy.loss[k, ] <- c(un.copy.loss[k]
+                                 , sum(tmp.probes[tmp.copy == un.copy.loss[k]]))
+                 mat.copy.loss <- rbind(c(0, sum(tmp.probes[tmp.copy >= 0]))
+                                         , mat.copy.loss)
+                 mat.copy.loss[, 2] <- mat.copy.loss[, 2]/sum(tmp.probes)
+                 return(mat.copy.loss)
+         }
+
+ }
> ################################################################
>
> fun.sample <- function(vals, perms){
+         mat.tmp <- matrix(NA, nr = perms, nc = length(vals))
+         for(k in 1:perms){
+                 if(k %% 1000 == 0)
+                         print(k)
+                 for(i in 1:length(vals))
+                         mat.tmp[k, i] <- sample(vals[[i]][, 1], 1, prob = vals[[i]][, 2])
+         }
+         return(mat.tmp)
+ }
> ################################################################
>
>
> fun.find.zeros <- function(x){
+         if(length(which(x == 0)) == length(which(!is.na(x))))
```

```
+                 return(FALSE)
+         else
+                 return(TRUE)
+
+ }
> ###############################################
> # Merge contiguous signififcant regions
> ###############################################
>
> fun.update.sig.regions <- function(sig.regions){
+         row.ind <- as.numeric(rownames(sig.regions))
+         start <- sig.regions[1, 2]
+         end <- sig.regions[1, 3]
+         chrom <- sig.regions[1, 1]
+         count <- 1
+         for(i in 2:length(row.ind)){
+                 if(sig.regions[i, 1] == sig.regions[i - 1, 1]
+                                         & row.ind[i] - row.ind[i -1] == 1){
+                         end[count] <- sig.regions[i, 3]
+                 }else{
+                         start <- c(start, sig.regions[i, 2])
+                         end <- c(end, sig.regions[i, 3])
+                         chrom <- c(chrom, sig.regions[i, 1])
+                         count <- count + 1
+                 }
+         }
+         return(cbind(chrom, start, end))
+
+ }
> ###################################################
> # Detect Genes in Significant Regions
> ###################################################
>
> fun.find.siggenes <- function(sig.regions, annot, biotype = "biotype"){
+         annot <- annot[annot[, biotype] == "protein_coding", ]
+         pos.tmp <- annot[, c("chromosome_name", "start_position", "end_position")]
+         pos.tmp <- apply(pos.tmp, 2, as.numeric)
+         genes.matrix <- matrix(NA, nr = 0, nc = ncol(annot))
+         for(i in 1:nrow(sig.regions)){
+          ww <- which( sig.regions[i, 1] == pos.tmp[, 1]
+                 & ( (pos.tmp[, 2] <= sig.regions[i, 3]
+                 &  pos.tmp[, 3] >= sig.regions[i, 3]  )
+                 |(pos.tmp[, 2] <= sig.regions[i, 2]
+                 &  pos.tmp[, 3] >= sig.regions[i, 2])
+                 |(pos.tmp[, 2] >= sig.regions[i, 2]
+                  & pos.tmp[, 3] <= sig.regions[i, 3]  )
```

```
+                 ))
+          genes.matrix <- rbind(genes.matrix, annot[ww, ])
+         }
+         return(genes.matrix)
+ }
> ############################################################
>
> function.plot.smooth <- function(segments, cent
+                         , scores.neg, scores.pos, lwd = 1
+                         , sigs.neg, sigs.pos, gene.neg, gene.pos, gain.sig.pos
+                         ,loss.sig.pos, smooth = TRUE
+                         , ylab, ydist = 0.3){
+    if(length(which(is.na(scores.neg))) == length(scores.neg))
+           scores.neg <- rep(NA, nrow(segments))
+   if(length(which(is.na(scores.pos))) == length(scores.pos))
+           scores.pos <- rep(NA, nrow(segments))
+   if(length(which(is.na(sigs.neg))) == length(sigs.neg))
+           sigs.neg <- rep(NA, nrow(segments))
+   if(length(which(is.na(sigs.pos))) == length(sigs.pos))
+           sigs.pos <- rep(NA, nrow(segments))
+
+   if(smooth == TRUE){
+            require(zoo)
+            scores.neg <- fun.smooth.score(scores.neg, segments)
+            #scores.pos <- fun.smooth.score(scores.pos, segments)
+
+   }
+   q.thresh.pos <- scores.pos[gain.sig.pos[1]]
+   q.thresh.neg <- scores.neg[loss.sig.pos[1]]
+
+   segments <- cbind(segments, scores.neg, scores.pos, sigs.neg, sigs.pos)
+
+   # Read centromere positions:
+   cent <- cent[, 2:4]
+   cent[, 1] <- substr(cent[, 1], 4, nchar(cent[, 1]))
+   cent <- centromere[as.numeric(cent[, 1]) %in% 1:22, ]
+   cent <- apply(cent, 2, as.numeric)
+
+   chrom.length <- fun.chrom.length(segments)
+   segments <- fun.add.chrom(segments, chrom.length)
+   cent <- fun.add.chrom(cent, chrom.length)
+
+   if(length(which(is.na(gene.neg))) != length(gene.neg))
+           gene.neg <- fun.add.chrom(gene.neg, chrom.length)
+   if(length(which(is.na(gene.pos))) != length(gene.pos))
+           gene.pos <- fun.add.chrom(gene.pos, chrom.length)
```

```
+
+   plot.matrix <- cbind( rep(segments[, 1], rep(2, nrow(segments)))
+                           , sort(c(segments[, 2], segments[, 3]))
+                           , rep(segments[, 4], rep(2, nrow(segments)))
+                           , rep(segments[, 5], rep(2, nrow(segments)))
+                           , rep(segments[, 6], rep(2, nrow(segments)))
+                           , rep(segments[, 7], rep(2, nrow(segments))))
+
+   colnames(plot.matrix) <- c("Chrom", "Position", "scores.neg"
+                           , "scores.pos", "sigs.neg", "sigs.pos")
+   plot.matrix[is.na(plot.matrix[, "scores.neg"]), "scores.neg"] <- 0
+   plot.matrix[is.na(plot.matrix[, "scores.pos"]), "scores.pos"] <- 0
+   plot.matrix[plot.matrix[, "scores.neg"] == 0, "sigs.neg"] <- NA
+   plot.matrix[plot.matrix[, "scores.pos"] == 0, "sigs.pos"] <- NA
+
+   if(length(which(is.na(scores.pos))) != length(scores.pos))
+           plot.matrix[plot.matrix[, "scores.neg"] > 0, "scores.neg"] <- 0
+   if(length(which(is.na(scores.pos))) != length(scores.pos))
+           plot.matrix[plot.matrix[, "scores.pos"] < 0, "scores.pos"] <- 0
+
+   # Assign test statistic values to significant genes
+   if(length(which(is.na(gene.neg))) != length(gene.neg)){
+           score.gene.neg <- c()
+           for(i in 1:nrow(gene.neg))
+                   score.gene.neg <- c(score.gene.neg
+                   , plot.matrix[which(plot.matrix[, "Position"] > gene.neg[i, 3])[1]
+                   , "scores.neg"])
+           gene.neg <- cbind(gene.neg, score.gene.neg)
+
+   }
+
+   if(length(which(is.na(gene.pos))) != length(gene.pos)){
+           score.gene.pos <- c()
+           for(i in 1:nrow(gene.pos))
+                   score.gene.pos <- c(score.gene.pos
+                   , plot.matrix[which(plot.matrix[, "Position"] > gene.neg[i, 3])[1]
+                   , "scores.pos"])
+           gene.pos <- cbind(gene.pos, score.gene.pos)
+
+   }
+
+
+           par(mar = c(5.1, 4.1, 4.1, 4.1))
+           plot(plot.matrix[, "Position"]
+                           , plot.matrix[, "scores.neg"], col = "red"
+                           , xaxs = "i", yaxs = "i", xaxt = "n"
```

```
+                         , ylim = c(min(plot.matrix[, "scores.neg"]
+                         , na.rm = T)- ydist, 0 )
+                         , type = "n", xlab = NA, ylab = NA
+                         , yaxt = "n")
+
+           axis(1, at = fun.chrom.mean(segments), labels = NA, las = 2, cex.axis = 0.8)
+
+           axis(1, at = fun.chrom.mean(segments), labels = as.character(1:22), las = 2
+           , cex.axis = 0.8, line = -0.3, tick = F)
+
+           axis(2, at = round(seq(min(plot.matrix[, "scores.neg"]
+                         , na.rm = T) , 0, 0.3)*10)/10
+                         , labels = NA
+                         , las = 1, cex.axis = 0.8)
+           mtext(ylab, side = 2, line = 2.3)
+
+           axis(2, at = round(seq(min(plot.matrix[, "scores.neg"]
+                         , na.rm = T) , 0, 0.3)*10)/10
+                         , labels = round(seq(min(plot.matrix[, "scores.neg"]
+                         , na.rm = T)
+                         , 0, 0.3)*10)/10
+                         , las = 1, cex.axis = 0.8, line = -0.3, tick = F)
+
+           for(i in seq(2 , 22, 2))
+            polygon(c(min(plot.matrix[plot.matrix[, "Chrom"] == i, "Position" ]
+             , na.rm = T)
+                   , max(plot.matrix[plot.matrix[, "Chrom"] == i, "Position" ]
+                   , na.rm = T)
+                   , max(plot.matrix[plot.matrix[, "Chrom"] == i, "Position" ]
+                   , na.rm = T)
+                   , min(plot.matrix[plot.matrix[, "Chrom"] == i, "Position" ]
+                   , na.rm = T))
+                   , c(min(plot.matrix[, "scores.neg"], na.rm = T) - ydist
+                   , min(plot.matrix[, "scores.neg"], na.rm = T) - ydist
+                   , 0 , 0 )
+                   , col = "#F0F0F0", border = NA)
+
+
+           abline(v = cent[, 2], lty = "dashed", lwd =0.1, col = "gray")
+
+
+           points(plot.matrix[, "Position"], plot.matrix[, "scores.neg"], type = "l"
+           , col = "blue", lwd = lwd)
+
+
+
```

```
+          if(length(which(is.na(gene.neg))) != length(gene.neg)){
+           for(i in 1:nrow(gene.neg))
+            lines(rep(gene.neg[i, 2], 1000)
+                  , seq(min(plot.matrix[, "scores.neg"]) - 0.1
+                  , min(plot.matrix[, "scores.neg"]) - ydist, length = 1000)
+                  , col = "green")
+          }
+          abline(h = 0, col = "black", lwd = lwd)
+          abline(h = q.thresh.neg, lty = "dashed", lwd = 0.1)
+          axis(4, c(q.thresh.neg), NA, las = 2)
+          axis(4, c(q.thresh.neg), c("Q = 0.25"), las = 2, tick = F, line = -0.3)
+          abline(h = c(min(plot.matrix[, "scores.neg"], na.rm = T) - ydist, 0))
+          abline(v = c(0, max(plot.matrix[, "Position"])))
+
+ }
> ##############################################################################
>
>
> fun.chrom.length <- function(seg){
+ # This function returns the length of chromosome 1,...,22
+ # seg: first three columns of the minimum conistent region matrix
+
+           chrom.length <- c()
+           for(i in 1:22){
+                   sub.chrom <- subset(seg, seg[, 1] == i)
+                   chrom.length <- c(chrom.length, sub.chrom[nrow(sub.chrom), 3])
+
+           }
+           return(chrom.length)
+
+   }
> ##############################################################################
>
> fun.add.chrom <- function(seg, chrom.length){
+ # This function adds the length of chromosome 1,...,(i - 1) to chromsomes 2,...,22
+ # seg: first three columns of the minimum conistent region matrix
+ # chrom.length: output of fun.chrom.length(...)
+          if(1 %in% seg[, 1])
+                  seg.tmp <-  subset(seg, seg[, 1] == 1)
+          else
+                  seg.tmp <- matrix(NA, nr = 0, nc = ncol(seg))
+          for(i in 2:22){
+                  if(!(i %in% seg[, 1]))
+                          next()
+                  sub.chrom <- subset(seg, seg[, 1] == i)
+                  sub.chrom[, 2] <- sub.chrom[, 2] + sum(chrom.length[1:(i - 1)])
```

```
+                 sub.chrom[, 3] <- sub.chrom[, 3] + sum(chrom.length[1:(i - 1)])
+                 seg.tmp <- rbind(seg.tmp, sub.chrom)
+         }
+         return(seg.tmp)
+
+ }
> ##############################################################################
>
> fun.smooth.score <- function(score, seg){
+ # This function applies smoothing (rollmean) to chromosome 1,...,22 separetely
+ # score: score to be smoothed (e.g. correlation)
+ # seg: Corresponding positions
+ # window: window size for rollmean
+ # The R package "fTrading" is required
+         require(fTrading)
+         score.tmp <- c()
+         for(i in 1:22)
+
+                 score.tmp <- c(score.tmp, rollMean(score[seg[, 1] == i]
+                 , n = round(length(score[seg[, 1] == i])/20)
+                 , trim = F, na.rm = TRUE))
+
+
+         return(score.tmp)
+
+
+ }
> ##############################################################################
>
> fun.chrom.mean <- function(seg){
+ # This function finds the middle of the positions for each chromosome.
+ # seg: Matrix with three columns, containing the positions.
+         chrom.mean <- c()
+         for(i in 1:22){
+                 sub.chrom <- subset(seg, seg[, 1] == i)
+                 chrom.mean <- c(chrom.mean
+                 , (min(sub.chrom[, 2]) + max(sub.chrom[, 3]))/2)
+
+         }
+         return(chrom.mean)
+
+ }
> ########################################
>
>
>
```

```
> fun.genes.copy <- function(x, gene.pos){
+
+
+         genes.copy <- matrix(NA, nr = nrow(gene.pos), nc = ncol(x) - 3)
+         for(i in 1:nrow(gene.pos)){
+             ww <- which( gene.pos[i, 1] == x[, 1]
+                                  &( (x[, 2] <= gene.pos[i, 3]
+                                  & x[, 3] >= gene.pos[i, 3])
+                                        |(x[, 2] <= gene.pos[i, 2]
+                                        & x[, 3] >= gene.pos[i, 2])
+                                      |(x[, 2] >= gene.pos[i, 2]
+                                      & x[, 3] <= gene.pos[i, 3])
+                                      ))
+             if(length(ww) > 1)
+               genes.copy[i, ] <- apply(x[ww, 4:ncol(x)], 2, median, na.rm = TRUE)
+             if(length(ww) == 1)
+               genes.copy[i, ] <- x[ww, 4:ncol(x)]
+         }
+
+         colnames(genes.copy) <- colnames(x)[4:ncol(x)]
+         rownames(genes.copy) <- rownames(gene.pos)
+         return(genes.copy)
+
+ }
> ##########################################################################
>
>
> fun.mapcopy.pos <- function(chrom.pos, seg){
+         vect.final <- rep(NA, length(unique(seg[, 1])))
+         names(vect.final) <- unique(seg[, 1])
+         sub <- subset(seg, as.numeric(seg[, 2]) == chrom.pos[1])
+         sub2 <- apply(sub[, 2:6], 2, as.numeric, na.rm = T)
+         ww <- which(((sub2[, 2] <= chrom.pos[3]
+                                 &  sub2[, 3] >= chrom.pos[ 3]  )
+                                 |(sub2[, 2] <= chrom.pos[2]
+                                  &  sub2[, 3] >= chrom.pos[2])
+                                 |(sub2[, 2] >= chrom.pos[2]
+                                  &  sub2[, 3] <= chrom.pos[3]  )
+                                 ))
+
+         sub.tmp <- sub[ww, ]
+         dups <-
+                 duplicated(sub.tmp[, 1]
+                 , fromLast = T) == T |
+                 duplicated(sub.tmp[, 1]) == T
+         n.dup <- sub.tmp[!dups, ]
```

```
+         vect.final[n.dup[, 1]] <- as.numeric(n.dup[, 6])
+         if(length(which(dups)) >= 1){
+          dup <- sub.tmp[dups, ]
+          val.dup <- rep(NA, length(unique(dup[, 1])))
+          names(val.dup) <- unique(dup[, 1])
+          for(kk in unique(dup[, 1])){
+           w.tmp <- which(dup[, 1] == kk)
+           tmp <- c()
+           for(ii in w.tmp)
+                 tmp <- c(tmp
+                         , pmax(c(chrom.pos[ 2], chrom.pos[3]))
+                         - pmin(c(as.numeric(dup[ii, 3])
+                         , as.numeric(dup[ii, 4]))))
+                   val.dup[kk] <- dup[w.tmp,  6][which(tmp == max(tmp))[1]]
+           }
+
+            vect.final[names(val.dup)] <- as.numeric(val.dup)
+         }
+         return(vect.final)
+ }
> ######################################################################
>
> fun.struct.complex <- function(sample, seg, thresh = 1e6){
+   vect.num.arm <- rep(0, length(unique(seg[, 2])))
+   names(vect.num.arm) <- as.numeric(unique(seg[, 2]))
+
+   sub.seg <- apply(subset(seg, seg[, 1] == sample)[, 2:6], 2, as.numeric, na.rm = TRUE)
+
+       for(chr in unique(sub.seg[, 1])){
+
+         sub <- subset(sub.seg, sub.seg[, 1] == chr)
+
+         mod.val <- as.numeric(names(which.max(table(rep(sub[, 5], sub[, 4])))))
+
+         chrom.length <- max(sub[, 3], na.rm = TRUE) - min(sub[, 2], na.rm = TRUE)
+         a <- sub[(sub[, 5] > mod.val | sub[, 5] < mod.val)
+                  & (sub[, 3] - sub[, 2]) > thresh, ]
+         if(length(a) == 6)
+           vect.num.arm[chr] <- 1
+         if(length(a) > 6)
+            vect.num.arm[chr] <- nrow(a)
+         }
+   return(vect.num.arm)
+
+ }
> ######################################################################
```

```
>
>
> fun.num.complex <- function(sample, seg, thresh = 0.25){
+   require(limma)
+   vect.whole.chrom <- rep(0, length(unique(seg[, 2])))
+   names(vect.whole.chrom) <- unique(seg[, 2])
+   sub.seg <- apply(subset(seg, seg[, 1] == sample)[, 2:6], 2, as.numeric, na.rm = TRUE)
+   ploidy <- weighted.median(sub.seg[, 5], w = sub.seg[, 3] - sub.seg[, 2], na.rm = TRUE)
+   sub.seg[, 5] <- sub.seg[, 5] - ploidy
+
+   for(chr in unique(sub.seg[, 1])){
+     sub <- subset(sub.seg, sub.seg[, 1] == chr)
+     tmp <- rep(sub[, 5], sub[, 4])
+     med.tmp <- median(tmp, na.rm = TRUE)
+     quant.tmp <- quantile(tmp, probs = c(thresh, 1 - thresh)  , na.rm = TRUE)
+     if(med.tmp < 0 & quant.tmp[2] < -0.8)
+         vect.whole.chrom[chr] <- abs(quant.tmp)
+     if(med.tmp > 0 & quant.tmp[1] > 0.8)
+         vect.whole.chrom[chr] <- abs(quant.tmp)
+
+   }
+   return(vect.whole.chrom)
+ }
> ######################################################################
>
> fun.fisher.mutations <- function(mut.status, cin.class){
+         n1 <- sum(mut.status == 1 & cin.class == "CIN(+)")
+         n2 <- sum(mut.status == 1 & cin.class == "CIN(-)")
+         n3 <- sum(mut.status == 0 & cin.class == "CIN(+)")
+         n4 <- sum(mut.status == 0 & cin.class == "CIN(-)")
+         mat.fisher <- matrix(c(n1, n2, n3, n4), nr = 2, nc = 2)
+         return(fisher.test(mat.fisher, alternative = "greater")$p.value)
+
+ }
> ######################################################################
>
> fun.fisher <- function(l1, l2, N){
+
+         both <- intersect(l1, l2)
+         mat.fisher <- matrix(NA, nr = 2, nc = 2)
+         mat.fisher[1, 1] <- length(both)
+         mat.fisher[2, 1] <- length(l1) - length(both)
+         mat.fisher[1, 2] <- length(l2) - length(both)
+         mat.fisher[2, 2] <-
+                         N - mat.fisher[1, 1] - mat.fisher[1, 2] -
+                         mat.fisher[2, 1]
```

```
+         return(fisher.test(mat.fisher))
+ }
> ########################################################################
>
> fun.hits.loss <- function(x)
+         return(length(which(x < 0)))
>
>
>
> ########################################################################
>
```

1.3 Load and Prepare Data

Load Data for General Use

```
> load("C:/Home/data_Sweave/Celllines/gene.annotation.HG18.RData")
> gene.annotation.HG18 <- as.matrix(gene.annotation.HG18)
> genes <- c("PIGN", "MEX3C", "ZNF516")
> gene.alias <- c("PIGN", "RKHD2", "ZNF516")
> genes <- cbind(genes
+                          , gene.annotation.HG18[
+                          match(genes, gene.annotation.HG18[, "external_gene_id"])
+                          , c("chromosome_name", "start_position", "end_position")])
> centromere <- read.table("C:/Home/data_Sweave/Celllines/centromere.txt"
+                 , header = FALSE)
> centromere <- as.matrix(centromere)
> cent.18q <- 16764896
```

Load and Prepare Sanger Cell Line Data

```
> load("C:/Home/data_Sweave/Celllines/cn.cl.data.RData")
> load("C:/Home/data_Sweave/Celllines/seg.colon.CL.RData")
> load("C:/Home/data_Sweave/Celllines/class.CL.RData")
> load("C:/Home/data_Sweave/Celllines/gii.weighted.colon.CL.RData")
> load("C:/Home/data_Sweave/Celllines/mat.mut.colon.CL.RData")
> load("C:/Home/data_Sweave/Celllines/expr.CL.rma.RData")
> cn.cl.data <- cn.cl.data[cn.cl.data[, 1] %in% 1:22, ]
> ploidy.colon.all.CL <- c()
> for(i in unique(seg.colon.CL[, 1])){
+         s = subset(seg.colon.CL, seg.colon.CL[, 1] == i)
+         ploidy.colon.all.CL <- c(ploidy.colon.all.CL
+         , weighted.median(as.numeric(s[, 6])
+         , w = as.numeric(s[, 4]) - as.numeric(s[, 3])  ))
+ }
> names(ploidy.colon.all.CL) <- unique(seg.colon.CL[, 1])
> mat.final.colon.CL <- function.matend(cn.cl.data)
```

```
> rownames(mat.final.colon.CL) <- 1:nrow(mat.final.colon.CL)
>
>
```

Load and Prepare Aneuploid Tumour data (BAC Arrays)

```
> load("C:/Home/data_Sweave/aneuploidTumors_BAC/segment.smoothed.CNA.object.RData")
> load("C:/Home/data_Sweave/aneuploidTumors_BAC/mat.final.gistic.RData")
> gistic.res <-
+   read.table(
+   "C:/Home/data_Sweave/aneuploidTumors_BAC/scores.gp_gistic.gistic.txt"
+   , sep = "\t", header = TRUE)
> gistic.res <- as.matrix(gistic.res)
> gistic.res[gistic.res[, 1] == "Del", "G.score"] <-
+         -as.numeric(gistic.res[gistic.res[, 1] == "Del", "G.score"])
> mat.pos.BAC <- mat.final.gistic[, 1:3]
> seg.BAC <- as.matrix(segment.smoothed.CNA.object$output)
> tmp <- apply(mat.pos.BAC, 1,fun.mapcopy.pos, seg = seg.BAC  )
> mat.final.BAC <- cbind(mat.pos.BAC, t(tmp))
```

Load and Prepare TCGA Tumour data

```
> load("C:/Home/data_Sweave/TCGA_Tumours/dup.colon.agilent.MA.RData")
> load("C:/Home/data_Sweave/TCGA_Tumours/mat.names.RData")
> load("C:/Home/data_Sweave/TCGA_Tumours/mat.final.colon.RData")
> load("C:/Home/data_Sweave/TCGA_Tumours/seg.mat.colon.GAP.RData")
> load("C:/Home/data_Sweave/TCGA_Tumours/mat.mut.colon.TCGA.RData")
> ploidy.colon.TCGA <- c()
> for(i in unique(seg.mat.colon.GAP[, 1])){
+         s = subset(seg.mat.colon.GAP, seg.mat.colon.GAP[, 1] == i)
+         ploidy.colon.TCGA <- c(ploidy.colon.TCGA
+                                         , weighted.median(as.numeric(s[, 6])
+                                         , w = as.numeric(s[, 4]) - as.numeric(s[, 3])))
+ }
> DNA.index.colon.TCGA <- c()
> for(i in unique(seg.mat.colon.GAP[, 1])){
+         s = subset(seg.mat.colon.GAP, seg.mat.colon.GAP[, 1] == i)
+         DNA.index.colon.TCGA <- c(DNA.index.colon.TCGA
+                                   , weighted.mean(as.numeric(s[, 6])
+                                   , w = as.numeric(s[, 4]) - as.numeric(s[, 3])))
+ }
> names(DNA.index.colon.TCGA) <- unique(seg.mat.colon.GAP[, 1])
> DNA.index.colon.TCGA <- DNA.index.colon.TCGA/2
> seg.mat.colon.GAP[, 2] <- trunc(as.numeric(seg.mat.colon.GAP[, 2]))
> names(ploidy.colon.TCGA) <- unique(seg.mat.colon.GAP[, 1])
> gii.weighted.colon.TCGA <- sapply(unique(seg.mat.colon.GAP[, 1])
+                           , gii.weight.chrom, seg = seg.mat.colon.GAP
```

```
+                         , ploidy = ploidy.colon.TCGA)
> tmp.final.colon <- mat.final.colon
> tmp.final.colon[, 4:ncol(tmp.final.colon)] <-
+                 sweep(tmp.final.colon[, 4:ncol(tmp.final.colon)]
+                 , MARGIN = 2, FUN = "-", ploidy.colon.TCGA)
>
>
>
>
```

2 Data Analysis

2.1 Cell Line Analysis

2.1.1 Identify Genes Located in CIN+ Loss Regions

```
> perms <- 1e4
> loss.values <- sapply(names(ploidy.colon.all.CL)
+                                          ,fun.find.prob.values
+                                          , seg = seg.colon.CL
+                                          , ploidy = ploidy.colon.all.CL
+                                          , gain = FALSE)
> tmp.perm.loss <- fun.sample(loss.values, perms)

[1] 1000
[1] 2000
[1] 3000
[1] 4000
[1] 5000
[1] 6000
[1] 7000
[1] 8000
[1] 9000
[1] 10000

> d.perms.loss <- d.stat(tmp.perm.loss, cl = class.CL, s0 = 0.5)$d
> tmp <- mat.final.colon.CL[, 4:ncol(mat.final.colon.CL)]
> tmp <- sweep(tmp, MARGIN = 2, FUN = "-", ploidy.colon.all.CL)
> tmp.loss <- tmp
> rm(tmp)
> tmp.loss[tmp.loss > 0] <- 0
> index.loss <- apply(tmp.loss, 1, fun.find.zeros)
> d.loss <- d.stat(tmp.loss, cl = class.CL, s0 = 0.5)$d
> d.loss[is.na(d.loss)] <- 0
> d.loss.tmp <- d.loss[index.loss]
> un.loss <- unique(d.loss.tmp)
```

```
> p.loss <- sapply(un.loss
+         , function(x) length(d.perms.loss[d.perms.loss <= x]))
> p.loss <- p.loss/length(d.perms.loss)
> p.loss.final <- rep(NA, length(d.loss.tmp))
> for(i in 1:length(un.loss))
+         p.loss.final[d.loss.tmp == un.loss[i]] <-  p.loss[i]
> q.loss <- qvalue(p.loss.final)$qvalue
> nam.loss <- as.numeric(rownames(tmp.loss[index.loss, ]))
> stat.loss.CL <- cbind(rep(0
+                         , nrow(mat.final.colon.CL))
+                         , rep(1, nrow(mat.final.colon.CL)))
> stat.loss.CL[nam.loss, 2] <- q.loss
> stat.loss.CL[nam.loss, 1] <- d.loss.tmp
> tmp.dip <- mat.final.colon.CL[, 4:ncol(mat.final.colon.CL)]
> num.dip.loss.cin <- apply(tmp.dip[, class.CL == 1], 1
+                         , function(x) length(which(x < 2)))
> num.dip.loss.stab <- apply(tmp.dip[, class.CL == 0], 1
+                         , function(x) length(which(x < 2)))
> num.loss.cin <- apply(tmp.loss[, class.CL == 1], 1
+                         , function(x) length(which(x < 0)))
> sig.loss <- mat.final.colon.CL[
+                         which(num.dip.loss.cin/ncol(tmp.dip[, class.CL == 1]) >= 0.3
+                                 & stat.loss.CL[, 2] < 0.25
+                                 & num.dip.loss.stab <= 1
+                                 & num.loss.cin >= 10), 1:3]
> sig.loss.update <- fun.update.sig.regions(sig.loss)
> genes.loss.CL <- fun.find.siggenes(sig.loss.update, gene.annotation.HG18)
> sig.loss.all <- mat.final.colon.CL[which(stat.loss.CL[, 2] < 0.25
+                 & num.loss.cin >= 10), 1:3]
> sig.loss.update.all <- fun.update.sig.regions(sig.loss.all)
> genes.loss.CL.all <- fun.find.siggenes(sig.loss.update.all
+                 , gene.annotation.HG18)
> print("Number Lost Genes (Loss vs Diploid Filter Not Included)")

[1] "Number Lost Genes (Loss vs Diploid Filter Not Included)"

> print(nrow(genes.loss.CL.all))

[1] 2383
```

2.1.2 Estimate Structural/Numerical Complexity

```
> struct.complex.CL <- sapply(unique(seg.colon.CL[, 1]), fun.struct.complex, seg.colon.CL)
> struct.score.CL <- colSums(struct.complex.CL, na.rm = TRUE)
> num.complex.CL <- sapply(unique(seg.colon.CL[, 1]), fun.num.complex, seg.colon.CL)
> num.score.CL <- colSums(abs(num.complex.CL), na.rm = TRUE)
> load("C:/Home/data_Sweave/TCGA_Tumours/struct.complex.TCGA.RData")
```

```
> load("C:/Home/data_Sweave/TCGA_Tumours/num.complex.TCGA.RData")
> #struct.complex.TCGA <- sapply(unique(seg.mat.colon.GAP[, 1])
> #        , fun.struct.complex, seg.mat.colon.GAP)
> struct.score.TCGA <- colSums(struct.complex.TCGA, na.rm = TRUE)
> #num.complex.TCGA <- sapply(unique(seg.mat.colon.GAP[, 1])
> #        , fun.num.complex, seg.mat.colon.GAP)
> num.score.TCGA <- colSums(abs(num.complex.TCGA), na.rm = TRUE)
>
>
```

2.2 BAC Array Analysis

```
> tmp.loss <- mat.final.BAC[, 4:ncol(mat.final.BAC)]
> tmp.loss[tmp.loss >= -0.1] <- 0
> num.loss <- abs(rowSums(sign(tmp.loss), na.rm = TRUE))
> q <- 10^(-as.numeric(gistic.res[, "X.log10.q.value."]))
> g.loss <-  max(as.numeric(gistic.res[gistic.res[, 1] == "Del" & q < 0.25, "G.score"]))
> rownames(mat.final.gistic) <- 1:nrow(mat.final.gistic)
> sig.loss <-
+         mat.final.gistic[which(mat.final.gistic[, "Del"] < g.loss & num.loss >= 13), 1:3]
> sig.loss.update <- fun.update.sig.regions(sig.loss)
> genes.loss.BAC <- fun.find.siggenes(sig.loss.update, gene.annotation.HG18)
> print("Number Significant Genes BAC:")

[1] "Number Significant Genes BAC:"

> print(nrow(genes.loss.BAC))

[1] 3027
```

3 Figures and Tables

3.1 Produce Figures

Figure 2a

```
> function.plot.smooth(mat.final.gistic[, 1:3]
+         ,  cent = centromere
+         , scores.neg = mat.final.gistic[, 5]
+         , scores.pos = NA
+         , lwd = 1
+         , sigs.neg = NA
+         , sigs.pos = NA
+         ,gain.sig.pos = NA
+         ,loss.sig.pos = which(mat.final.gistic[, 5]
+                 == max(as.numeric(gistic.res[gistic.res[, 1] == "Del"
+                 & q < 0.25, "G.score"])))[1]
+         , gene.neg = NA
+         , gene.pos = NA
+         , smooth = FALSE
+         , ylab = "GISTIC Score", ydist = 0.1)
>
>
```

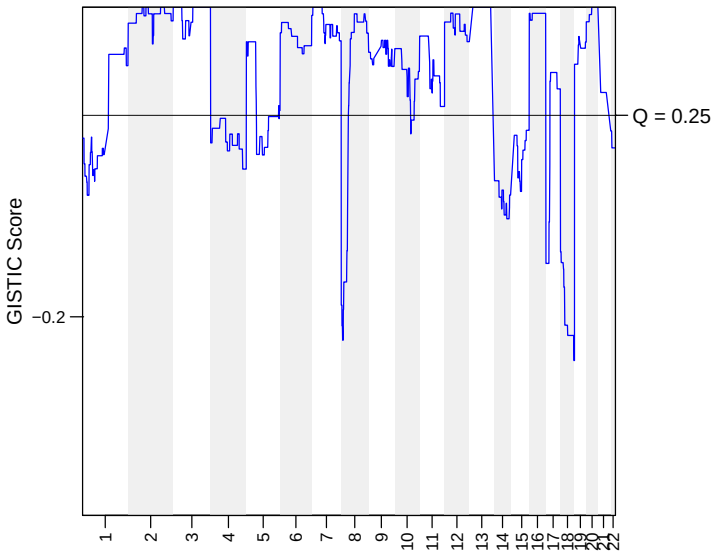


Figure 2b

```
> function.plot.smooth(mat.final.colon.CL[, 1:3]
+          ,  cent = centromere
+          , scores.neg = stat.loss.CL[, 1]
+          , scores.pos = NA
+          , lwd = 1
+          , sigs.neg = NA
+          , sigs.pos = NA
+          ,gain.sig.pos = NA
+          ,loss.sig.pos = which(stat.loss.CL[, 1]
+                                == max(stat.loss.CL[stat.loss.CL[, 2] < 0.25, 1]
+                                , na.rm = TRUE))
+          , gene.neg = apply(genes.loss.CL[genes.loss.CL[, "chromosome_name"] == 18
+          , c("chromosome_name", "start_position", "end_position")], 2, as.numeric)
+          , gene.pos = NA
+
+          , smooth = TRUE
+          , ylab = "d-Score", ydist = 0.3)
>
```

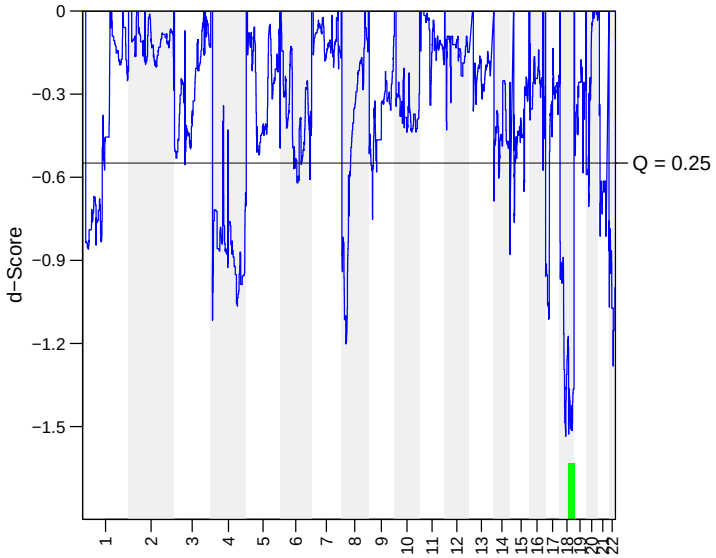
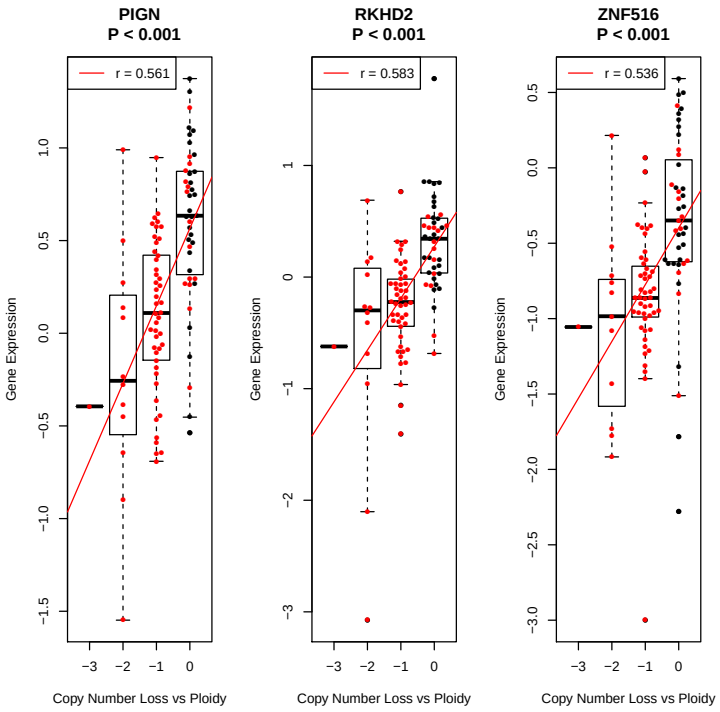


Figure 2e

```
> gene.pos <- apply(genes[, 2:ncol(genes)], 2, as.numeric)
> gene.copy <- fun.genes.copy(tmp.final.colon, gene.pos)
> rownames(gene.copy) <- genes[, 1]
> int <- intersect(colnames(gene.copy), colnames(dup.colon.agilent.MA))
> gene.copy <- gene.copy[, int]
> dup.colon.agilent.MA <- dup.colon.agilent.MA[, int]
> tmp.match <- match(colnames(gene.copy), names(gii.weighted.colon.TCGA))
> class.TCGA <- gii.weighted.colon.TCGA[tmp.match] < 0.2
> par(mfrow = c(1, 3))
> for(gene in rownames(gene.copy)){
+  gene.expr <- dup.colon.agilent.MA[
+                 mat.names[which(mat.names[, 2] == gene), 1]
+                 , ]
+  if(length(which(mat.names[, 2] == gene)) > 1){
+          cor.tmp <- c()
+          for(i in 1:nrow(gene.expr))
+         cor.tmp <- c(cor.tmp, cor(gene.expr[i, ]
+                                   , gene.copy[gene, ]
+                                  , method = "spearman"
```

```
+                                , use = "pairwise.complete.obs"))
+
+          index <- which.max(cor.tmp)
+          cor.copy.expr <- round(1000*cor(gene.expr[index,
+                                ], gene.copy[gene, ], method = "spearman"
+                                , use = "pairwise.complete.obs"))/1000
+          p.copy.expr <- round(1000*cor.test(gene.expr[index, ]
+                                          , gene.copy[gene, ]
+                                          , method = "spearman")$p.value)/1000
+          boxplot(gene.expr[index, ] ~ gene.copy[gene, ]
+                          , ylab = "Gene Expression"
+                          , xlab = "Copy Number Loss vs Ploidy"
+                         , main = paste(gene.alias[genes == gene], "\n", "P < 0.001"
+                         , sep = "\t") )
+          legend("topleft", paste("r = ", cor.copy.expr,sep = "")
+                          , lty  =1 , col = "red")
+          beeswarm(gene.expr[index, ] ~ gene.copy[gene, ], cex = 0.8
+                          , add = TRUE, pch = 16
+                          , pwcol = ifelse(class.TCGA, "black", "red"))
+          abline(lm(gene.expr[index, ]
+                          ~ as.numeric(as.factor((gene.copy[gene, ])))$coefficients
+                          , col = "red")
+  }else{
+         cor.copy.expr <- round(1000*cor(gene.expr, gene.copy[gene, ]
+                                          , method = "spearman"
+                                          , use = "pairwise.complete.obs"))/1000
+         p.copy.expr <- round(1000*cor.test(gene.expr
+                                          , gene.copy[gene,
+                                          ], method = "spearman")$p.value)/1000
+
+         boxplot(gene.expr ~ gene.copy[gene, ]
+                                , ylab = "Gene Expression"
+                                , xlab = "Copy Number Loss vs Ploidy"
+                                , main = paste(gene.alias[genes == gene]
+                                , "\n", "P < 0.001"
+                                , sep = "\t") )
+         legend("topleft", paste("r = ", cor.copy.expr,sep = "")
+                          , lty  =1 , col = "red")
+         beeswarm(gene.expr ~ gene.copy[gene, ], cex = 0.8
+                          , add = TRUE, pch = 16
+                          , pwcol = ifelse(class.TCGA, "black", "red"))
+         abline(lm(gene.expr ~
+                         as.numeric(as.factor((gene.copy[gene, ])))$coefficients
+                          , col = "red")
+         }
+ }
```

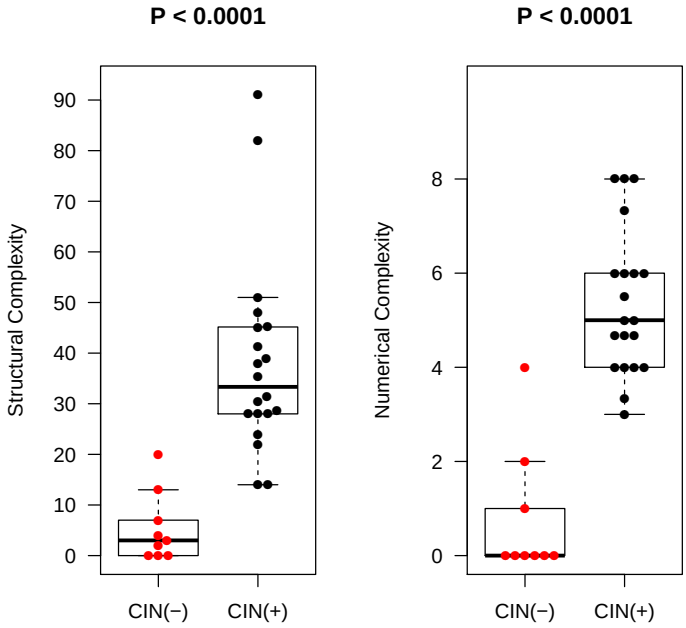


3.2 Supplementary Figures

Supplementary Figure 1a and 1b

```
> MIN <- c(4, 5, 6, 7, 10, 13, 18, 21, 26)
> col <- rep("black", 29)
> col[MIN] <- "red"
> CINMIN <- rep("CIN", 29)
> CINMIN[MIN] <- "MIN"
> wilcox.test(struct.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN)

        Wilcoxon rank sum test with continuity correction

data:  struct.score.CL/(ploidy.colon.all.CL/2) by CINMIN
W = 178, p-value = 3.638e-05
alternative hypothesis: true location shift is not equal to 0

> wilcox.test(num.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN)

        Wilcoxon rank sum test with continuity correction

data:  num.score.CL/(ploidy.colon.all.CL/2) by CINMIN
```

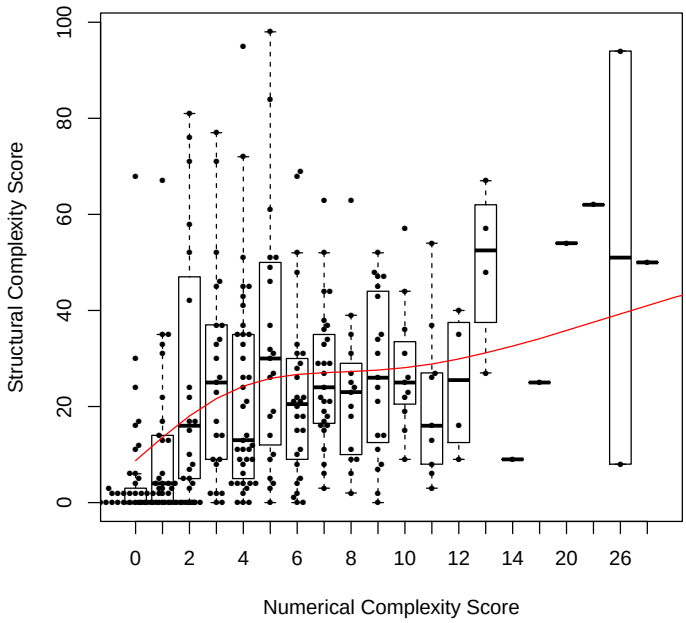
```
W = 176, p-value = 4.747e-05
alternative hypothesis: true location shift is not equal to 0

>

> par(mfrow = c(1, 2))
> boxplot(struct.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN
+         , ylab = "Structural Complexity", yaxt = "n", outline = F
+         , names = c("CIN(+)", "CIN(-)")
+         , ylim = c(0, max(struct.score.CL/(ploidy.colon.all.CL/2)) + 2)
+         , at = c(2, 1), main = "P < 0.0001")
> axis(2, seq(0, max(struct.score.CL/(ploidy.colon.all.CL/2)), 10)
+         , seq(0,max(struct.score.CL/(ploidy.colon.all.CL/2)), 10)
+         , las = 2)
> beeswarm(struct.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN
+         , add = TRUE, pch = 16, pwcol = col, at = c(2, 1))
> boxplot(num.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN
+         , ylab = "Numerical Complexity", yaxt = "n", outline = F
+         , names = c("CIN(+)", "CIN(-)")
+         , ylim = c(0, max(num.score.CL/(ploidy.colon.all.CL/2)) + 2)
+         , at = c(2, 1), main = "P < 0.0001")
> axis(2, seq(0, max(num.score.CL/(ploidy.colon.all.CL/2)), 2)
+         , seq(0,max(num.score.CL/(ploidy.colon.all.CL/2)), 2
+         ), las = 2)
> beeswarm(num.score.CL/(ploidy.colon.all.CL/2) ~ CINMIN
+         , add = TRUE, pch = 16, pwcol = col, at = c(2, 1))
```



Supplementary Figure 1c

```
> struct.score.norm.TCGA <- struct.score.TCGA/(ploidy.colon.TCGA/2)
> num.score.norm.TCGA <- num.score.TCGA/(ploidy.colon.TCGA/2)
> fit.norm<- sreg(round(num.score.norm.TCGA),round(struct.score.norm.TCGA))
> print(cor(num.score.norm.TCGA, struct.score.norm.TCGA, method = "spearman"))

[1] 0.4554851

> print(cor.test(num.score.norm.TCGA, struct.score.norm.TCGA, method = "spearman"))

        Spearman's rank correlation rho

data:  num.score.norm.TCGA and struct.score.norm.TCGA
S = 2600268, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4554851

>
```

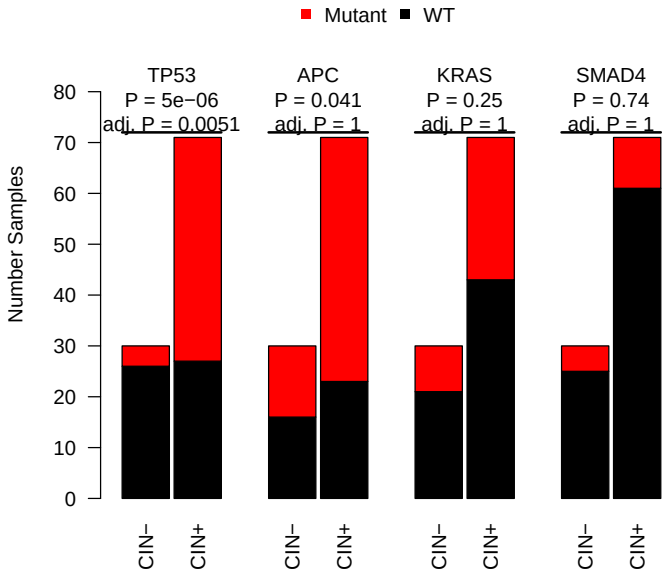
```
> boxplot(round(struct.score.norm.TCGA) ~ round(num.score.norm.TCGA)
+         , outline = F
+         , ylab = "Structural Complexity Score"
+         , xlab = "Numerical Complexity Score")
> beeswarm(round(struct.score.norm.TCGA) ~ round(num.score.norm.TCGA)
+         , add = TRUE, pch = 16, cex = 0.6)
> points(fit.norm$predicted$x + 1, fit.norm$predicted$y
+         , col = "red", type = "l")
>
```



Supplementary Figure 6a

```
> mut.match <- match(colnames(mat.mut.colon.TCGA), names(gii.weighted.colon.TCGA))
> class.mut.TCGA <- gii.weighted.colon.TCGA[mut.match] > 0.2
> p.fisher.mut.TCGA <- apply(mat.mut.colon.TCGA, 1, fun.fisher.mutations
+                  , cin.class = ifelse(class.mut.TCGA, "CIN(+)", "CIN(-)"))
> p.adj.mut.TCGA <- mt.rawp2adjp(p.fisher.mut.TCGA, proc = "BH")
> p.adj.mut.TCGA <- p.adj.mut.TCGA$adjp[order(p.adj.mut.TCGA$index), ]
> rownames(p.adj.mut.TCGA) <- names(p.fisher.mut.TCGA)
> tt1 <- table(mat.mut.colon.TCGA["TP53", ]
+                         , ifelse(class.mut.TCGA
+                                             , "CIN(+)", "CIN(-)"))
> tt2 <- table(mat.mut.colon.TCGA["APC", ]
+                         , ifelse(class.mut.TCGA
+                                             , "CIN(+)", "CIN(-)"))
> tt3 <- table(mat.mut.colon.TCGA["KRAS", ]
+                         , ifelse(class.mut.TCGA
+                                             , "CIN(+)", "CIN(-)"))
> tt4 <- table(mat.mut.colon.TCGA["SMAD4", ]
+                         , ifelse(class.mut.TCGA
+                                             , "CIN(+)", "CIN(-)"))
```

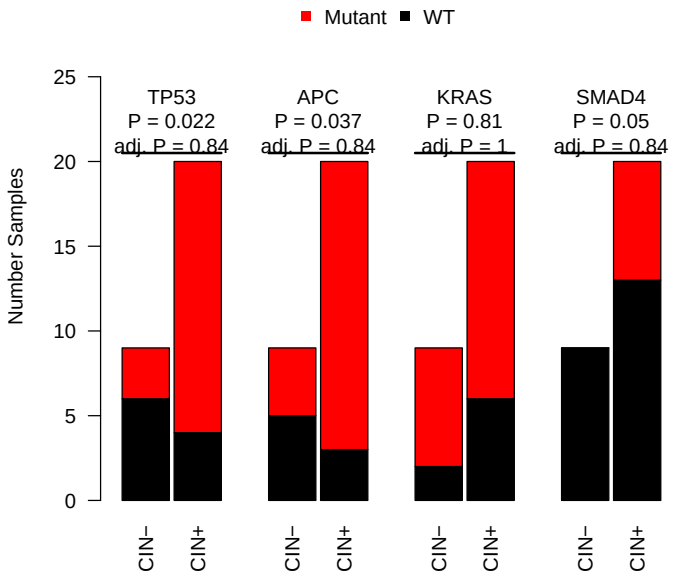
```
>

> b <- barplot(cbind(tt1, tt2, tt3, tt4), col = c("black", "red")
+                 , ylab = "Number Samples", ylim = c(0, 100)
+                 , yaxt = "n", xlab = NA, las = 2
+                 , space = c(0.1, 0.1, 1, 0.1, 1, 0.1, 1, 0.1)
+                 , names  = rep(c("CIN-", "CIN+"), 4))
>         legend("top", c("Mutant", "WT"), col = c("red", "black")
+                 , pch = 15, bty = "n", horiz = TRUE)
>         axis(2, seq(0, 80, 10), seq(0, 80, 10), las = 2)
>         points(c(b[1] - 0.5, b[2] + 0.5), c(max(colSums(tt1)) + 1
+                 , max(colSums(tt1)) + 1)
+                 , type = "l", lwd = 2)
>         text((b[1] + b[2])/2, max(colSums(tt1)) + 7.5
+                 , paste("TP53\n", "P = "
+                 , signif(p.fisher.mut.TCGA["TP53"], 2), "\nadj. P = "
+                 ,signif(p.adj.mut.TCGA["TP53", 2], 2) ,sep = ""))
>         points(c(b[3] - 0.5, b[4] + 0.5), c(max(colSums(tt2)) + 1
+                 , max(colSums(tt2)) + 1)
+                 , type = "l", lwd = 2)
>         text((b[3] + b[4])/2, max(colSums(tt2)) + 7.5
+                 , paste("APC\n", "P = "
+                 , signif(p.fisher.mut.TCGA["APC"], 2), "\nadj. P = "
+                 ,signif(p.adj.mut.TCGA["APC", 2], 2), sep = ""))
>         points(c(b[5] - 0.5, b[6] + 0.5), c(max(colSums(tt3)) + 1
+                 , max(colSums(tt3)) + 1)
+                 , type = "l", lwd = 2)
>         text((b[5] + b[6])/2, max(colSums(tt3)) + 7.5
+                 , paste("KRAS\n", "P = "
+                 , signif(p.fisher.mut.TCGA["KRAS"], 2), "\nadj. P = "
+                 ,signif(p.adj.mut.TCGA["KRAS", 2], 2), sep = ""))
>         points(c(b[7] - 0.5, b[8] + 0.5), c(max(colSums(tt4)) + 1
+                 , max(colSums(tt4)) + 1)
+                 , type = "l", lwd = 2)
>         text((b[7] + b[8])/2, max(colSums(tt4)) + 7.5
+                 , paste("SMAD4\n", "P = "
+                 , signif(p.fisher.mut.TCGA["SMAD4"], 2), "\nadj. P = "
+                 ,signif(p.adj.mut.TCGA["SMAD4", 2], 2), sep = ""))
>
```



Supplementary Figure 6b

```
> MIN <- c(4, 5, 6, 7, 10, 13, 18, 21, 26)
> cin.class <- rep("CIN(+)", 29)
> cin.class[MIN] <- "CIN(-)"
> p.fisher.mut.CL <- apply(mat.mut.colon.CL, 1, fun.fisher.mutations
+                  , cin.class = cin.class)
> p.adj.mut.CL <- mt.rawp2adjp(p.fisher.mut.CL, proc = "BH")
> p.adj.mut.CL <- p.adj.mut.CL$adjp[order(p.adj.mut.CL$index), ]
> rownames(p.adj.mut.CL) <- names(p.fisher.mut.CL)
> tt1 <- table(mat.mut.colon.CL["TP53", ], cin.class)
> tt2 <- table(mat.mut.colon.CL["APC", ], cin.class)
> tt3 <- table(mat.mut.colon.CL["KRAS", ], cin.class)
> tt4 <- table(mat.mut.colon.CL["SMAD4", ], cin.class)
>
>

> b <- barplot(cbind(tt1, tt2, tt3, tt4), col = c("black", "red")
+                  , ylab = "Number Samples", ylim = c(0, 30)
+                  , yaxt = "n", xlab = NA, las = 2
+                  , space = c(0.1, 0.1, 1, 0.1, 1, 0.1, 1, 0.1)
```

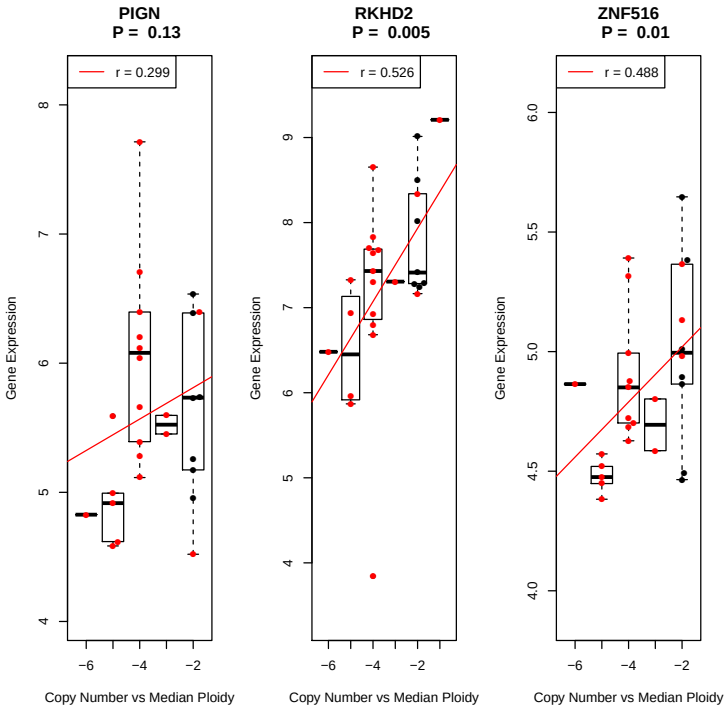
```
+                , names  = rep(c("CIN-", "CIN+"), 4))
>        legend("top", c("Mutant", "WT"), col = c("red", "black")
+                , pch = 15, bty = "n", horiz = TRUE)
>        axis(2, seq(0, 25, 5), seq(0, 25, 5), las = 2)
>        points(c(b[1] - 0.5, b[2] + 0.5), c(max(colSums(tt1)) + .5
+                , max(colSums(tt1)) + .5)
+                , type = "l", lwd = 2)
>        text((b[1] + b[2])/2, max(colSums(tt1)) + 2.4
+                , paste("TP53\n", "P = "
+                , signif(p.fisher.mut.CL["TP53"], 2), "\nadj. P = "
+                ,signif(p.adj.mut.CL["TP53", 2], 2) ,sep = ""))
>        points(c(b[3] - 0.5, b[4] + 0.5), c(max(colSums(tt2)) + .5
+                , max(colSums(tt2)) + .5)
+                , type = "l", lwd = 2)
>        text((b[3] + b[4])/2, max(colSums(tt2)) + 2.4
+                , paste("APC\n", "P = "
+                , signif(p.fisher.mut.CL["APC"], 2), "\nadj. P = "
+                ,signif(p.adj.mut.CL["APC", 2], 2), sep = ""))
>        points(c(b[5] - 0.5, b[6] + 0.5), c(max(colSums(tt3)) + .5
+                , max(colSums(tt3)) + .5)
+                , type = "l", lwd = 2)
>        text((b[5] + b[6])/2, max(colSums(tt3)) + 2.4
+                , paste("KRAS\n", "P = "
+                , signif(p.fisher.mut.CL["KRAS"], 2), "\nadj. P = "
+                ,signif(p.adj.mut.CL["KRAS", 2], 2), sep = ""))
>        points(c(b[7] - 0.5, b[8] + 0.5), c(max(colSums(tt4)) + .5
+                , max(colSums(tt4)) + .5)
+                , type = "l", lwd = 2)
>        text((b[7] + b[8])/2, max(colSums(tt4)) + 2.4
+                , paste("SMAD4\n", "P = "
+                , signif(p.fisher.mut.CL["SMAD4"], 2), "\nadj. P = "
+                ,signif(p.adj.mut.CL["SMAD4", 2], 2), sep = ""))
>
```



Supplementary Figure 8a

```
> xx2 <- as.list(hthgu133aALIAS2PROBE)
> affy.ids <- unlist(xx2[genes])
> tmp <- mat.final.colon.CL[, 4:ncol(mat.final.colon.CL)]
> tmp <- sweep(tmp, MARGIN = 2, FUN = "-", ploidy.colon.all.CL)
> mat.final.colon.CL <- cbind(mat.final.colon.CL[, 1:3], tmp)
> gene.pos <- gene.pos <- apply(genes[, 2:ncol(genes)], 2, as.numeric)
> gene.copy <- fun.genes.copy(mat.final.colon, gene.pos)
> gene.copy.CL <- fun.genes.copy(mat.final.colon.CL, gene.pos)
> rownames(gene.copy.CL) <- genes[, 1]
> gene.copy.CL[gene.copy.CL > 0] <- 0
> MIN <- c(4, 5, 6, 7, 10, 13, 18, 21, 26)
> class.cl <- rep("CIN", 29)
> class.cl[MIN] <- "MIN"
> par(mfrow = c(1, 3))
> for(gene in genes[, 1]){
+         gene.expr.CL <- expr.CL.rma[affy.ids[gene], -c(5, 21)]
+         cor.copy.expr.CL <- round(1000*cor(gene.expr.CL
+                 , gene.copy.CL[gene, -c(5, 21)], method = "spearman"
+                 , use = "pairwise.complete.obs"))/1000
```

```
+         p.copy.expr.CL <- round(1000*cor.test(gene.expr.CL
+                                 , gene.copy.CL[gene, -c(5, 21)]
+                                 , method = "spearman")$p.value)/1000
+         boxplot(gene.expr.CL ~ gene.copy.CL[gene, -c(5, 21)]
+                 , ylab = "Gene Expression", xlab = "Copy Number vs Median Ploidy"
+                 , main = paste(gene.alias[genes[, 1] == gene], "\n", "P = "
+                 , p.copy.expr.CL ,sep = "\t") , ylim = c(min(gene.expr.CL) - .5
+                 , max(gene.expr.CL) + .5))
+         legend("topleft", paste("r = ", cor.copy.expr.CL,sep = "")
+                 , lty  =1 , col = "red")
+         beeswarm(gene.expr.CL ~ gene.copy.CL[gene, -c(5, 21)]
+                 , add = TRUE, pch = 16
+                 , pwcol = ifelse(class.cl[-c(5, 21)] == "CIN"
+                 , "red", "black"))
+         abline(lm(gene.expr.CL ~
+         as.numeric(as.factor((gene.copy.CL[gene, -c(5, 21)]))))$coefficients
+         , col = "red")
+ }
>
>
>
```


PIGN
P = 0.13
r = 0.299
Gene Expression
Copy Number vs Median Ploidy
RKHD2
P = 0.005
r = 0.526
Gene Expression
Copy Number vs Median Ploidy
ZNF516
P = 0.01
r = 0.488
Gene Expression
Copy Number vs Median Ploidy