1 Automated image processing pipeline for quantifying tumor cellular contents

Here we describe our image processing pipeline implemented as a R package CRImage which generates objective and quantitative scores of cellular contents directly from H&E stained images. The pipeline combines unsupervised classification algorithms with methods for incorporating pathological prior knowledge.

Our primary goal for the image processing pipeline is to obtain cell-specific information from H&E stained images. H&E is the most commonly used stain for the morphological assessment of tissues, where nuclei are predominantly stained by haematoxylin (dark blue to black), whilst cytoplasm and connective tissue is stained by eosin (pink)\(^1\).

While freezing tumors helps preserve high-quality genetic material, it significantly complicates image analysis. Frozen sections are more likely to incur a number of artifacts including reduced resolution, variation in staining intensity, folded tissue, air bubbles, shattering of tissue and incomplete sectioning due to over-freezing\(^2\). Therefore, it is necessary to purposely develop an image processing pipeline for our goal.

The code for our image processing pipeline is freely available as a R package CRImage as a contribution to the open source R platform for statistical computing\(^3\). The pipeline classifies segmented objects into four categories of “cancer cell”, “stromal cell”, “lymphocyte” and “artifact” based on a training set provided by a pathologist. The entire process consists of six steps: pre-processing, segmentation, feature extraction, classification, kernel smoothing, and a hierarchical refinement step.

In brief, the pipeline works as the follows (with details in the following sections). During pre-processing, original images were converted into standard format, divided into sub-images, cropped if necessary, and scanned for artifacts. Cell nuclei were segmented from which features such as diameter, texture and shape parameters are obtained. These sets of features were subjected to a classifier which distinguishes the four classes. The outcome of the classification was then inserted into a spatial smoothing routine to account for local neighbors. Finally, a novel hierarchical model was employed to yield final classification result by incorporating global knowledge of the image.

1.1 Data pre-processing

The original images from the METABRIC database are kept in the svs file format. Each of these images was then divided into 2,000 × 2,000 pixel sub-images in CWS format using Digital Slide Studio (Aperio) where the quality parameter was set to 80 (maximum). The sub-images were converted to greyscale afterwards for segmentation and classification.

1.2 Image segmentation

The first coarse segmentation was carried out using our own implementation of Otsu’s histogram thresholding\(^4\). To cope with staining artifacts, i.e. cluttered areas of high dye concentration which are not nucleus-shaped, we applied morphological opening\(^5\) using a disc-shaped structuring element to retain only nuclei, some of which were still clustered. These remaining clustered nuclei were separated by distance transformation and divided by the watershed algorithm using tolerance and extenstion parameters of \(^6\).
1.3 Feature extraction

For every nucleus 93 phenotypical features were obtained using functions from R package EBImage\(^7\) (Table S1): 
13 Hull features, 13 Haralick features, 18 image moments and 49 Zernicke moments. Hull features include size of the nucleus, perimeter, eccentricity and acircularity\(^7\). Haralick features describe nucleus texture and are calculated from the co-occurrence matrix of the grey values\(^8\). Image moments characterize the two dimensional density distribution function of the grey values of an image\(^9\). Zernike moments are shape parameters invariant to translation and rotation based on Zernike polynomial\(^10\). In addition, we added a spatial feature to account for the number of neighbors within a distance of 50 pixels. Hence, there are a total of 94 features for classification.

1.4 Support Vector Machine classification

All 94 features were subjected to classification using a Support Vector Machine (SVM)\(^11\) with a radial basis function (RBF) kernel\(^12\). The SVM was trained on four classes of cells using R-package e1071\(^13\). The gamma parameter of the RBF kernel was calculated by \(1/k\) with \(k\) being the number of features. The RBF kernel outperformed both polynomial and linear kernels in terms of cross validation accuracy.

An initial training set for the SVM was created by a pathologist on a cell-by-cell basis, which consists of 238 cancer cells, 102 lymphocytes, 216 stromal cells, and 315 artifacts from 7 sub-images). Using this initial training set as a starting point, we then constructed site-specific training sets by a semi-automated adaptation approach. For each sample set, a sub-image with good classification outcome with the initial training set was manually selected and added to the initial set, obtaining a final set of 4,329 cells (cancer: 2120, lymphocyte: 858, stromal: 848, artifact: 552) in the discovery set. This approach both speeds up the training process and substantially increases prediction power, and therefore is recommended for analyzing any new sample set since there is no universally effective training set.

1.5 Kernel smoothing

We used a kernel smoother to adjust the SVM classification outcome of adjacent nuclei. This smoothing step is crucial. Pathologists not only classify a cell by their visual appearance but also take into account other cells in the spatial proximity. This is important for tumor section images as different cell types can appear similar due to crushed cells during sectioning, as exemplified in Figure 2A. We therefore adjusted the class probabilities of a cell by the class probabilities of the surrounding cells. The intuition is that a cell is likely to be misclassified if it is classified as stromal cell with low confidence but also surrounded by cells classified as cancer with high confidence.

We implemented the kernel smoother using a kernel smoother\(^14\) with a tri-cube link function. The new class probability of a nucleus was estimated by calculating the new class probability for a nucleus \(x_0\) by:

\[
\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i)y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)},
\]

where \(y_i\) is the SVM output class probability as determined by the singular classifier and the sum runs over all nuclei in the image \((x_i)\). The kernel function \(K_\lambda\) is defined as:

\[
K_\lambda(x_0, x) = D\left(\frac{||x - x_0||_2}{\lambda}\right),
\]

\(||x - x_0||_2\) is the Euclidean distance of \(x\) and \(y\) coordinates and

\[
D(t) = \begin{cases} 
(1 - |t|^3)^3 & \text{if } |t| \leq 1 \\
0 & \text{otherwise},
\end{cases}
\]

is the tri-cube function\(^13\). The parameter \(\lambda\) defines the window size and was set to 50 pixels (average diameter of a cancer cell is 10 pixels). We performed smoothing for all four class probabilities from the SVM classifier and a cell is assigned to the class with highest probability. Cells labeled as lymphocytes by the SVM were excluded from smoothing, as the spatial dependency does not hold for lymphocytes which can be infiltrated into tumors. After smoothing, if a cell has the highest probability to be artifact or lymphocyte, its class remains unchanged, because we do not assume spatial gathering for these cells.

1.6 A hierarchical model providing a multi-resolution view of tumor patterns

We developed a hierarchical model to be applied immediately following the previous steps. Although kernel smoothing can correct individual misclassification cases, it loses power when the majority of the cases are
misclassified due to quality reasons such as staining variability. An additional step to correct for the misclassifications is therefore necessary in this scenario.

A common observation in pathology is that cancer cells tend to appear in clusters, such as the example shown in Figure 1. By incorporating this prior knowledge, we designed a multi-resolution model to extract cancer cell clusters and correct on results from the previous steps. To remove unnecessary details, we first scaled down the sub-images by a factor of eight, then applied color correction (detailed below) and Gaussian blurring. Since details are shielded in lower resolution images, tumor patterns emerge as can be seen in Figure 2a. Then, the cancer clusters were extracted with a new pixel classifier (details to follow), followed by morphological opening, smoothing of the objects, and removing artifacts. Finally, the locations of cancer clusters were projected down to the high resolution images in a hierarchical manner. Consequently, cells located in these clusters were termed as cancer cells, if and only if they were previously classified as artifacts or stromal cells. Since lymphocytes can infiltrate into cancer clusters, the model does not apply to lymphocytes. This multi-resolution, hierarchical information flow enables pattern recognition in a manner close to human perception.

The working pipeline of the hierarchical model is depicted (Figure 2). Figure 2 shows a case where cancer cells were misclassified as artifacts because of the irregular segmentation due to staining variability. When the patterns from its shrunked image are superimposed onto the original images, cancer cells in these clusters can be identified, while the artifacts in folded tissue remain. This model is particularly useful for identifying large number of cancer cells misclassified as artifacts, as illustrated in Figure 2. Notably, this model can distinguish cancer cell clusters from real artifacts such as the folded tissues in Figure 2.

**Color correction** In order to maintain color consistency for all images, color standardization to training images was applied. Training images and the sub-images were both converted to the $l'\alpha'\beta'$ color space, which minimizes correlation between color channel. In particular, the sub-images were standardized to the training images by:

$$
l' = (l_o - \langle l_o \rangle) \frac{\sigma_l}{\sigma'_l} + \langle l_t \rangle,
$$

$$\alpha' = (\alpha_o - \langle \alpha_o \rangle) \frac{\sigma_{\alpha}}{\sigma'_{\alpha}} + \langle \alpha_t \rangle,$$

$$\beta' = (\beta_o - \langle \beta_o \rangle) \frac{\sigma_{\beta}}{\sigma'_{\beta}} + \langle \beta_t \rangle,$$

(1)
where \( \langle I_o \rangle \) is the mean value of the \( l \) channel of the original sub-image, \( \langle I_t \rangle \) is the mean of the training images, \( \sigma_o \) and \( \sigma_t \) are the standard deviations for the original and training images, respectively. The same holds for the \( \alpha \) and \( \beta \) channel of the image. After standardizing, the images were converted back to the RGB color space (Figure 2b).

A pixel classifier A support vector machine was trained on the RGB values of the cancer clusters and the stroma for classifying every pixel into two classes: stroma and cancer cell cluster (Figure 2c-d). The training set included 6,070 pixel of the stroma and 3,813 pixel of cancer cell cluster. We reached an accuracy of 98.0\% in 10-fold cross validation. Cells in the cancer cell clusters previously labeled as stromal cells or artifacts were assigned to class ‘cancer cell”. Nothing was changed for cancer cells and lymphocytes.

2 Image analysis for the METABRIC data: Discovery set

To run CRImage on the METABRIC whole section tumor section images, use the “run.CRImage.R” file in the Script folder. This script was generated for processing all images in parallel on a cluster. For each sample, the Aperio image has been split into 50-300 subimages of 2,000-by-2,000 pixels depending on the size of the sections in the image. In each of the cluster run, subimages of the same sample are processed one after another. At the end, the script also computes the locations of cells in the whole images based on the position of each subimages by running “run.ImageSpatial.R”, and the spatial K-score for stromal cells by “run.kstats.R” in the same folder.

3 Reproducing analysis results in the paper

Load the CRImage run results for the Aperio images in a cohort “allData.rdata”. This object contains sample ID, patient clinical information, and image analysis results from CRImage, including “cancer” (percentage of cancer...
cells), “lym” (percentage of lymphocytes), “stromal” (percentage of stromal cells), “kstats.stromal” (K-scores for the stromal cell spatial pattern), etc. We then source essential functions for analysis in ‘functions.R’.

```r
> load("./Data/allData.rdata")
> source("./Script/Functions.R")
```

To plot the triangle plot of the distribution of the cellular contents (as in Figure 2D), we need the variables ‘stromal’, ‘lym’, and ‘cancer’:

```r
> library(ade4)
> D <- allData
> attach(D)
> D.plot <- D[,c("stromal","lym","cancer")]
> pdf("Figure2.pdf",width=7,height=4)
> par(mfrow=c(1,2))
> my.triangle.plot(D.plot, scale=TRUE, col="black", cpoint=1.5)
> my.triangle.plot(D.plot, scale=FALSE, col="black", cpoint=1.5, show.position=FALSE)
> dev.off()
```

null device
1

For the trend test for correlation between image-based cellularity scores and pathological cellularity scores, we used package SAGx. There are three variables we use, “TumourAreaRatio” (cancer cell number to the tissue area in pixels), “cancer” (number of cancer cells to all cells), and “lymphocyte” (number of lymphocytes to all cells). The first two are correlated to “cellularity” the pathological scores for tumor cellularity, and the last one is correlated to “LIPathScore”.

```r
> library(SAGx)
> D <- allData
> if (!is.factor(D$cellularity))
+ D$cellularity <- as.factor(D$cellularity)
> D$cellularity <- relevel(relevel(D$cellularity,"moderate"),"low")
> my.scale <- function(foo,q1=0,q2=1){
+ foo2 <- (foo - quantile(foo,q1))/(quantile(foo,q2)-quantile(foo,q1))
+ foo2[foo2<0]<-0
+ return(foo2)
+ foo2[foo2>1]<-1
+ }
> ourLI <- lym
> ourCell <- cancer
> ourCellArea <- my.scale(TumourAreaRatio)
> pathoCell <- D$cellularity
```
To validate our correction algorithm, we used Signal-to-Noise-Ratio data before and after correction, a FISH experimental data, and a dilution series data.

> pdf("Figure4.pdf",width=7,height=7)
> par(mfrow=c(2,2))
> ## Panel B
> load("./Data/Signal2Noise.rdata")
> upperlim <- max(all.res[,2:3])
> plot(all.res$Orig,all.res$Corr,col="#00000077",las=1,pch=19,xlab="Original signal-to-noise",ylab="Correction")
> null device
1

To validate our correction algorithm, we used Signal-to-Noise-Ratio data before and after correction, a FISH experimental data, and a dilution series data.
```r
> abline(0,1)
> if ("FISH_SN6_HER2_score.csv"%in% dir("./Data")){
+ E <- read.csv("./Data/FISH_SN6_HER2_score.csv")
+ E$cat[E$cat==4] <- 3
+ attach(E)
+ y <- log2(E$CombinedISHscore/2)
+ x1 <- E$SNP6bc
+ x2 <- E$SNP6ac
+ h1b=E$SNP6bc
+ h1a=E$SNP6ac
+ h1=h1a
+ h1=1*(h1 > log2(3/2)) + 1*(h1 > log2(6/2))+ 1*(h1> log2(4/2))
+ h1a=h1
+ h1b=h1
+ h1=1*(h1 > log2(3/2)) + 1*(h1 > log2(6/2))+ 1*(h1> log2(4/2))
+ h1b=h1
+
+ plot(NA,xlim=c(.5,6.5),ylim=c(-.5,5),xaxs="i",las=1,ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n")
+ for (xx in c(0,4)) polygon(x=c(xx+.5,xx+.5,xx+2.5,xx+2.5),y=c(-3,6,6,-3),col="lightgrey",border=NA)
+ abline(h=0)
+ boxplot(h1 ~ cat, at=seq(1,5,by=2)+.05, col="purple", add=TRUE, xaxt="n",yaxt="n",varwidth=FALSE)
+ boxplot(h1b ~ cat, at=seq(2,6,by=2)-.05 , col="orange", add=TRUE, xaxt="n",yaxt="n",varwidth=FALSE)
+ axis(1,at=seq(1,5,by=2)+.5,label=c("neutral","low amp","high amp"))
+ legend("topleft",col=c("purple","orange"),legend=c("Before correction","After correction"),pch=c(19,19), inset=0.01, cex=.7)
+ text(x=seq(1,9,by=2)+.5,y=-.4,labels=table(cat))
+ segments(x0=1.05, x1=5.05, y0=3.25)
+ segments(x0=1.05,y0=3.25, y1=3.05)
+ segments(x0=5.95, y0=3.25, x1=5.95)
+ segments(x0=1.95, x1=5.95, y0=3.75)
+ segments(x0=1.95, y0=3.75, y1=3.55)
+ segments(x0=5.95, y0=3.75, y1=3.55)
+ text(x=3.05, y=3.25, paste("p=" , signif(t.test(h1b[cat==0], h1b[cat==2])$p.value,1)), pos=3)
+ text(x=3.95, y=3.75, paste("p=" , signif(t.test(h1a[cat==0], h1a[cat==2])$p.value,1)), pos=3)
+ }
> if ("ROCDATA.RData"%in%dir("./Data")){
+ load("./Data/ROCDATA.RData")
+ SensCorr2 <- rbind(SensCorr,matrix(NA,ncol=ncol(SensCorr),nrow=nrow(Sens)-nrow(SensCorr)))
+ boxplot( cbind(Sens,SensCorr2)[,rep(1:9,each=2)+c(0,9)],col=c("orange","purple"),main="Sensitivity")
+ SpecCorr2 <- rbind(SpecCorr,matrix(NA,ncol=ncol(SpecCorr),nrow=nrow(Spec)-nrow(SpecCorr)))
+ boxplot( cbind(Spec,SpecCorr2)[,rep(1:9,each=2)+c(0,9)],col=c("orange","purple"),main="Specificity")
+ }
> graphics.off()
>
> subset <- allData$er="neg" & (!is.na(allData$er)) & (!is.na(allData$LIPathScore))
> pdf("Figure5A.pdf",width=7,height=4.5)
> par(mfrow=c(1,2))
> plotSurv(allData$S[subset], (allData$lym[subset]>0.08) +1, fileType="", name="Image")
```

```r
[1] 0.02011363
```

```r
> plotSurv(allData$S[subset], allData$LIPathScore[subset],fileType="",col=1:3, name="Pathology")
```

```r
[1] 0.6396448
```

```r
> dev.off()
```

null device

1
```
Figure 5: Copy number data correction results as in Figure 4

Figure 6: Comparing survival stratification power for two lymphocytic groupings as in Figure 5A
To correlate lymphocyte abundance with survival, we used the proportion of lymphocytes and breast cancer specific deaths. Then we built a SVM classifier using R package kernlab integrating both expression and image-based lymphocyte scores.

```R
> library(cluster)
> library(survival)
> library(kernlab)
> lymERNeg <- allData$lym[allData$er=="neg"]
> S.ERNeg <- Surv(allData$S[allData$er=="neg",1], allData$S[allData$er=="neg",2])
> LIExprERNeg <- allData$LIscore[allData$er=="neg"]
> thrsh <- 0.08
> clusters <- 1+(lymERNeg<=thrsh)*1
> ## correlation between LIExprScore and LymRatio
> cor(LIExprERNeg, lymERNeg,method="spearman",use="complete")
[1] 0.2281304
> cor(LIExprERNeg[(lymERNeg>thrsh)],lymERNeg[(lymERNeg>thrsh)],method="spearman",use="complete")
[1] 0.3512097
> pdf(file="Figure5B.pdf",width=4.5,height=4)
> ERNegSample <- as.character(allData$Sample[allData$er=="neg"])
> S.ERNeg <- allData$S[match(ERNegSample, allData$Sample),]
> LI.ERNeg <- allData$LIscore[match(ERNegSample, allData$Sample)]
> lymERNeg <- allData$lym[match(ERNegSample, allData$Sample)]
> y <- S.ERNeg[,2]
> dat <- data.frame(LI.ERNeg=LI.ERNeg,lymERNeg=lymERNeg, y=y)
> x <- data.frame(expr=LI.ERNeg,lym=lymERNeg)
> library(kernlab)
> set.seed(11)
> mm <- ksvm(y~LI.ERNeg+lymERNeg, data=dat, type="C-svc", kernel = "rbfdot", cross=10)
> mm@cross
[1] 0.1533333
> my.plot.ksvm(mm,data=x,xlim=c(-0.1,.5),ylim=c(-1.5,3.5), grid=100,main=paste("SVM classification accuracy",mm@cross*100, "%"))
null device
  1
> subset <- allData$er=="neg" & (!is.na(allData$er)) & (!is.na(allData$LIscore))
> LIExprNeg <- allData$LIscore[subset]
> LIclustersERNeg <- LIExprNeg > quantile(allData$LIscore,2/3, na.rm=T)
> dat <- data.frame(LIExprERNeg=LIExprNeg,lymERNeg=lymERNeg, y=1-allData$S[subset,2])
> cc1 <- NULL
> for (i in 1:20){
+  mm <- ksvm(y~lymERNeg, data=dat, type="C-svc", cross=10)
+  cc1 <- c(cc1, mm@cross)
+ }```

Using automatic sigma estimation (sigest) for RBF or laplace kernel

```R
> mm@cross
[1] 0.1533333
> my.plot.ksvm(mm,data=x,xlim=c(-0.1,.5),ylim=c(-1.5,3.5), grid=100,main=paste("SVM classification accuracy",mm@cross*100, "%"))
null device
  1
> subset <- allData$er=="neg" & (!is.na(allData$er)) & (!is.na(allData$LIscore))
> lymERNeg <- allData$lym[subset]
> LIExprERNeg <- allData$LIscore[subset]
> LIclustersERNeg <- LIExprERNeg > quantile(allData$LIscore,2/3, na.rm=T)
> dat <- data.frame(LIExprERNeg=LIExprERNeg,lymERNeg=lymERNeg, y=1-allData$S[subset,2])
> cc1 <- NULL
> for (i in 1:20){
+  mm <- ksvm(y~lymERNeg, data=dat, type="C-svc", cross=10)
+  cc1 <- c(cc1, mm@cross)
+ }```

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel

Using automatic sigma estimation (sigest) for RBF or laplace kernel
Figure 7: Integrating image and expression data as in Figure 5B

```r
> 1-mean(cc1)
[1] 0.7441667

> cc2 <- NULL
> for (i in 1:20){
+  mm <- ksvm(y~LIExprERNeg, data=dat, type="C-svc", cross=10)
+  cc2 <- c(cc2, mm@cross)
+ }
```
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel

> 1-mean(cc2)
[1] 0.6696667

> cc <- NULL; M <- NULL
> for (i in 1:20){
+ mm <- ksvm(y~LIExprERNeg+lymERNeg, data=dat, type="C-svc", cross=10, verbose=FALSE)
+ cc <- c(cc, mm@cross)
+ M <- c(M, list(mm))
+ }
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Using automatic sigma estimation (sigest) for RBF or laplace kernel

> 1-mean(cc)
[1] 0.8575

> pdf("Figure5C.pdf",width=10,height=5)
> par(mfrow=c(1,3))
> set.seed(92)
> mm <- ksvm(y~LIExprERNeg+lymERNeg, data=dat, type="C-svc", cross=10, verbose=FALSE)

Using automatic sigma estimation (sigest) for RBF or laplace kernel

> plotSurv(allData$S[subset], allData$lym[subset]>0.08, fileType="", lty=1:2, col=1:2, name="Image")
[1] 0.02011363

> plotSurv(allData$S[subset], LIclustersERneg, fileType="", col=c("blue", "orange"), lty=1:2, name="Expr")
[1] 0.7392802

> plotSurv(allData$S[subset][!is.na(allData$S[subset,2])], mm@fitted, fileType="", lty=1:2, name="Integrated")
[1] 1.181948e-10

> dev.off()
null device
1
Figure 8: Comparing lymphocyte proportions, expression signature, and the integrated classifier as in Figure 5C

```r
> library(Hmisc)
> dat <- allData[, "kstats.stromal"]
> S <- allData$S
> deaths <- allData$deaths
> set <- as.character(allData$er) == "neg"
> dat.s <- dat[set]
> deaths.s <- deaths[set]
> pdf("Figure6A.pdf", width=4, height=4)
> ## distribution of K-stats
> hist(dat.s, main="", col="lightgrey", border=NA, xlab="K-stat discovery data", las=1)
> abline(v=dat.s[deaths.s==1])
> abline(v=mean(dat, na.rm=TRUE), col="red", lwd=2, lty="dashed")
> abline(v=quantile(dat.s, c(.25, .75), na.rm=TRUE), col="red", lwd=2)
> dev.off()

null device
1

> library(Hmisc)
> set <- allData$er == "neg" & !is.na(allData$er)
> auc <- allData$kstats.stromal
> pdf("Figure6B.pdf", width=10, height=10)
> par(mfrow=c(2,2))
> cl <- q3(auc, 0.25, 0.75)[set] + 1
> plotSurv(allData$S[set], cl, fileType="", col = 1:3)
> errbar(x=((1e+6)*allData[, "nOthers"] / allData$Area)[set], y=auc[set], yplus=auc[set]+allData$kstats.stromal.sd[set]/2, yminus=auc[set]-allData$kstats.stromal.sd[set]/2, add=TRUE)
> points(((1e+6)*allData[, "nOthers"] / allData$Area)[set], auc[set], bg=cl, col=1, pch=allData$deaths[set] + 23, cex=allData$deaths[set] + 1)
> cl[cl==3] <- 1
> plotSurv(allData$S[set], cl, fileType="", col = 1:2, lty=1:2)
> dev.off()

[1] 0.08067881

> plot(((1e+6)*allData[, "nOthers"] / allData$Area)[set], auc[set], col=1, main=colnames(auc)[i], xlab="C")
> errbar(x=((1e+6)*allData[, "nOthers"] / allData$Area)[set], y=auc[set], yplus=auc[set]+allData$kstats.stromal.sd[set]/2, yminus=auc[set]-allData$kstats.stromal.sd[set]/2, add=TRUE)
> points(((1e+6)*allData[, "nOthers"] / allData$Area)[set], auc[set], bg=cl, col=1, pch=allData$deaths[set] + 23, cex=allData$deaths[set] + 1)
> dev.off()

[1] 0.02493811
```
Figure 9: Distribution of K-scores as in Figure 6A
Figure 10: Survival curves and distributions for the stromal spatial patterns, before and after two extreme groups were merged, as in Figure 6BC
1 Reproducing analysis results: Validation set

Load the CRImage run results for the Aperio images in a cohort “allData.rdata”. This object contains sample ID, patient clinical information, and image analysis results from CRImage, including “cancer” (percentage of cancer cells), “lym” (percentage of lymphocytes), “stroml” (percentage of stromal cells), “kstats.stromal” (K-scores for the stromal cell spatial pattern), etc. We then source essential functions for analysis in ‘functions.R’.

```r
> load("../Data/allData.rdata")
> source("../Script/functions.R")
```

To plot the triangle plot of the distribution of the cellular contents (as in Figure 2D), we need the variables ‘stromal’, ‘lym’, and ‘cancer’:

```r
> library(ade4)
> D <- allData
> attach(D)
> D.plot <- D[,c("stromal","lym","cancer")]
> pdf("Figure2.pdf",width=7,height=4)
> par(mfrow=c(1,2))
> my.triangle.plot(D.plot, scale=TRUE, col="black", cpoint=1.5)
> my.triangle.plot(D.plot, scale=FALSE, col="black", cpoint=1.5, show.position=FALSE)
> dev.off()
```

null device
1

For the trend test for correlation between image-based cellularity scores and pathological cellularity scores, we used package SAGx. There are three variables we use, “TumourAreaRatio” (cancer cell number to the tissue area in pixels), “cancer” (number of cancer cells to all cells), and “lymphocyte” (number of lymphocytes to all cells). The first two are correlated to “cellularity” the pathological scores for tumor cellularity, and the last one is correlated to “LIPathScore”.

```r
> library(SAGx)
> D <- allData
> if (!is.factor(D$cellularity))
```

![Figure 1: Distributions of cellular contents for all samples, as in manuscript Figure 2D](image)

Figure 1: Distributions of cellular contents for all samples, as in manuscript Figure 2D
To validate our correction algorithm, we used Signal-to-Noise-Ratio data before and after correction, a FISH experimental data, and a dilution series data.
> abline(0,1)
> if ("FISH_SNP6_HER2_score.csv" %in% dir("./Data")){
+ E <- read.csv("./Data/FISH_SNP6_HER2_score.csv")
+ E$cat[E$cat==4] <- 3
+ attach(E)
+ y <- log2(E$CombinedISHscore/2)
+ x1 <- E$SNP6bc
+ x2 <- E$SNP6ac
+ h1b=E$SNP6bc
+ h1a=E$SNP6ac
+ h1=h1a
+ h1=1*(h1 > log2(3/2)) + 1*(h1 > log2(6/2))+ 1*(h1> log2(4/2))
+ h1a=h1
+ h1=h1b
+ h1=1*(h1 > log2(3/2)) + 1*(h1 > log2(6/2))+ 1*(h1> log2(4/2))
+ h1b=h1
+ plot(NA,xlim=c(.5,6.5),ylim=c(-.5,5),xaxs="i",las=1,xlab="Copy-number levels based on SNP 6.0",ylab="Aberration status based on FISH",xaxt="n")
+ abline(h=0)
+ boxplot(h1b ~ cat, at=seq(1,5,by=2)+.05, col="purple", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ boxplot(h1a ~ cat, at=seq(2,6,by=2)-.05 , col="orange", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ abline(h=0)
+ boxplot(h1b ~ cat, at=seq(1,5,by=2)+.05, col="purple", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ boxplot(h1a ~ cat, at=seq(2,6,by=2)-.05 , col="orange", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ abline(h=0)
+ segments(x0=1.05, x1=5.05, y0=3.25)
+ segments(x0=1.05,y0=3.25, y1=3.55)
+ segments(x0=5.05,y0=3.25, y1=3.55)
+ segments(x0=1.95, x1=5.95, y0=3.75)
+ segments(x0=1.95, y0=3.75, y1=3.55)
+ segments(x0=5.95, y0=3.75, y1=3.55)
+ text(x=3.05, y=3.25, paste("p=", signif(t.test(h1b[cat==0], h1b[cat==2])$p.value,1)), pos=3)
+ axes(xaxt="n")
+ boxplot(h1a ~ cat, at=seq(1,5,by=2)-.05 , col="orange", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ abline(h=0)
+ boxplot(h1b ~ cat, at=seq(1,5,by=2)+.05, col="purple", add=TRUE, xaxt="n",ylab="Copy-number levels based on SNP 6.0",xlab="Aberration status based on FISH",xaxt="n",varwidth=FALSE)
+ abline(h=0)
+ segments(x0=1.05, x1=5.05, y0=3.25)
+ segments(x0=1.05,y0=3.25, y1=3.55)
+ segments(x0=5.05,y0=3.25, y1=3.55)
+ segments(x0=1.95, x1=5.95, y0=3.75)
+ segments(x0=1.95, y0=3.75, y1=3.55)
+ segments(x0=5.95, y0=3.75, y1=3.55)
+ text(x=3.05, y=3.25, paste("p=", signif(t.test(h1b[cat==0], h1b[cat==2])$p.value,1)), pos=3)
+ text(x=3.95, y=3.75, paste("p=", signif(t.test(h1a[cat==0], h1a[cat==2])$p.value,1)), pos=3)
+ }
> if ("ROCDATA.RData" %in% dir("./Data")){
+ load("./Data/ROCDATA.RData")
+ boxplot(cbind(SensCorr,SensCorr2)[,rep(1:9,each=2)+c(0,9)],col=c("orange","purple"),main="Sensitivity")
+ boxplot(cbind(Spec,SpecCorr2)[,rep(1:9,each=2)+c(0,9)],col=c("orange","purple"),main="Specificity")
+ }
> graphics.off()

> subset <- allData$er=="neg" & (!is.na(allData$er)) & (!is.na(allData$LIPathScore))
> pdf("Figure5.pdf",width=7,height=4.5)

Figure 2: Correlation with pathological scores as in Figure 3
Figure 3: Copy number data correction results as in Figure4
To correlate lymphocyte abundance with survival, we used the proportion of lymphocytes and breast cancer specific deaths. Then we built a SVM classifier using R package kernlab integrating both expression and image-based lymphocyte scores.

```r
> library(cluster)
> library(survival)
> library(kernlab)
> lymERNeg <- allData$lym[allData$er=="neg"]
> S.ERNeg <- Surv(allData$S[allData$er=="neg",1], allData$S[allData$er=="neg",2])
> LIExprERNeg <- allData$LIscore[allData$er=="neg"]
> thrsh <- 0.08
> clusters <- 1+(lymERNeg<=thrsh)*1
> cor(LIExprERNeg, lymERNeg,method="spearman",use="complete")
```

Figure 4: Comparing survival stratification power for two lymphocytic groupings as in Figure 5
Figure 5: Comparing survival stratification power for two lymphocytic groupings as in Figure 5A.

\begin{verbatim}
> cor(LIExprERNeg[(lymERNeg>thrsh)], lymERNeg[(lymERNeg>thrsh)], method="spearman", use="complete")

[1] 0.3992095

> pdf(file="Figure5B.pdf", width=4.5, height=4)
> ERNegSample <- as.character(allData$Sample[allData$er=="neg"])
> S.ERNeg <- allData$S[match(ERNegSample, allData$Sample),]
> LI.ERNeg <- allData$LIscore[match(ERNegSample, allData$Sample),]
> lymERNeg <- allData$lym[match(ERNegSample, allData$Sample)]
> y <- S.ERNeg[,2]
> dat <- data.frame(LI.ERNeg=LI.ERNeg, lymERNeg=lymERNeg, y=y)
> x <- data.frame(expr=LI.ERNeg, lym=lymERNeg)
> library(kernlab)
> set.seed(11)
> mm <- ksvm(y~LI.ERNeg+lymERNeg, data=dat, type="C-svc", kernel = "rbfdot", cross=10)
> mm@cross

[1] 0.43

> my.plot.ksvm(mm, data=x, xlim=c(-0.1,.5), ylim=c(-1.5,3.5), grid=100, main=paste("SVM classification accuracy", mm@cross*100, ",&sqlfig;"))
> dev.off()

Using automatic sigma estimation (sigest) for RBF or laplace kernel

> mm@cross

[1] 0.43

> subset <- allData$er=="neg" & (!is.na(allData$er)) & (!is.na(allData$LIscore))
> lymERNeg <- allData$lym[subset]
> LIExprERNeg <- allData$LIscore[subset]
> LIclustersERNeg <- LIExprERNeg > quantile(allData$LIscore, 2/3, na.rm=T)
> dat <- data.frame(LIExprERNeg=LIExprERNeg, lymERNeg=lymERNeg, y=1-allData$S[subset,2])
> ccl <- NULL
> for (i in 1:20){
+   mm <- ksvm(y~lymERNeg, data=dat, type="C-svc", cross=10)
+   ccl <- c(ccl, mm@cross)
+ }
\end{verbatim}
Figure 6: Integrating image and expression data as in Figure 5B

```r
> 1-mean(cc1)
[1] 0.5215

> cc2 <- NULL
> for (i in 1:20)
+  mm <- ksvm(y~LIExprERNeg, data=dat, type="C-svc", cross=10)
+  cc2 <- c(cc2, mm@cross)
+ }

> 1-mean(cc1)
[1] 0.5215
```
Using automatic sigma estimation (sigest) for RBF or laplace kernel

> 1-mean(cc2)
[1] 0.539

> cc <- NULL; M <- NULL
> for (i in 1:20) {
+   mm <- ksvm(y~LIExprERNeg+lymERNeg, data=dat, type="C-svc", cross=10, verbose=FALSE)
+   cc <- c(cc, mm@cross)
+   M <- c(M, list(mm))
+ }

Using automatic sigma estimation (sigest) for RBF or laplace kernel

> 1-mean(cc)
[1] 0.53225

> pdf("Figure5C.pdf", width=10, height=5)
> par(mfrow=c(1,3))
> set.seed(92)
> mm <- ksvm(y~LIExprERNeg+lymERNeg, data=dat, type="C-svc", cross=10, verbose=FALSE)

Using automatic sigma estimation (sigest) for RBF or laplace kernel

> plotSurv(allData$S[subset], allData$lym[subset]>0.08, fileType="", lty=1:2, col=1:2, name="Image")
[1] 0.0375917

> plotSurv(allData$S[subset], LIclustersERneg, fileType="", col=c("blue", "orange"), lty=1:2, name="Expr")
[1] 0.01709962

> plotSurv(allData$S[subset][!is.na(allData$S[subset,2])], mm@fitted, fileType="", lty=1:2, name="Integ..."
Figure 7: Comparing lymphocyte proportions, expression signature, and the integrated classifier as in Figure 5C

[1] 7.380975e-05

> dev.off()
null device

1

> library(Hmisc)
> dat <- allData[, "kstats.stromal"]
> S<-allData$S
> deaths <- allData$deaths
> set <- as.character(allData$er) == "neg"
> dat.s <- dat[set]
> deaths.s <- deaths[set]
> pdf("Figure6A.pdf", width=4, height=4)
> ## distribution of K-stats
> hist(dat.s,main="",col="lightgrey",border=NA,xlab="K-stat discovery data",las=1)
> abline(v=dat.s[deaths.s==1])
> abline(v=mean(dat,na.rm=TRUE),col="red",lwd=2,lty="dashed")
> abline(v=quantile(dat.s,c(.25,.75),na.rm=TRUE),col="red",lwd=2)
> dev.off()
null device

1

> library(Hmisc)
> set <- allData$er == "neg" & !is.na(allData$er)
> auc <- allData$kstats.stromal
> pdf("Figure6BC.pdf", width=10, height=10)
> par(mfrow=c(2,2))
> cl <- q3(auc,0.25,0.75)[set]+1
> plotSurv(allData$S[set], cl, fileType="",col=1:3)

[1] 0.07484347

> plot(((1e+6)*allData[, "nOthers"] / allData$Area)[set], auc[set], col=1, main=colnames(auc)[i], xlab="Cell/Area ratio", lty=1)
> pch=allData$deaths[set]
> errbar(x=((1e+6)*allData[, "nOthers"] / allData$Area)[set], y=auc[set], yplus=auc[set]+allData$kstats.sd[set], yminus=auc[set]-allData$kstats.sd[set])
> cl[cl==3] <- 1
> plotSurv(allData$S[set], cl, fileType="",col=1:2, lty=1:2)
Figure 8: Distribution of K-scores as in Figure 6A

[1] 0.02332142

> plot(((1e+6) * allData[, "nOthers"] / allData$Area)[set], auc[set], col=1, main=colnames(auc)[i], xlab="Cell/Area ratio", ylab="K-stats", cex.lab=1.5)
> errbar(x=((1e+6) * allData[, "nOthers"] / allData$Area)[set], y=auc[set], yplus=auc[set]+allData$kstats.stromal.sd[set]/2, yminus=auc[set]-allData$kstats.stromal.sd[set]/2, add=TRUE)
> points(((1e+6) * allData[, "nOthers"] / allData$Area)[set], auc[set], bg=cl, col=1, pch=allData$deaths[set]+23, cex=allData$deaths[set]+1)
> dev.off()
Figure 9: Survival curves and distributions for the stromal spatial patterns, before and after two extreme groups were merged, as in Figure 6BC.
References


Table 1: Representative nuclei features grouped into their categories as used in our SVM classifier.

<table>
<thead>
<tr>
<th>Topological features</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>area number of pixel</td>
<td></td>
</tr>
<tr>
<td>p perimeter</td>
<td></td>
</tr>
<tr>
<td>pdm mean distance from the center to perimeter</td>
<td></td>
</tr>
<tr>
<td>pdsd standard deviation of the distance to perimeter</td>
<td></td>
</tr>
<tr>
<td>effr radius of a circle with the same area</td>
<td></td>
</tr>
<tr>
<td>acirc fraction of pixels outside of the circle with radius effr</td>
<td></td>
</tr>
<tr>
<td>sf shape factor, $\frac{\text{area}}{2\sqrt{(\text{area})}}$</td>
<td></td>
</tr>
<tr>
<td>theta hull orientation angle, in radians</td>
<td></td>
</tr>
<tr>
<td>l1 largest eigenvalue of the covariance matrix</td>
<td></td>
</tr>
<tr>
<td>l2 lowest eigenvalue of the covariance matrix</td>
<td></td>
</tr>
<tr>
<td>ecc eccentricity, equals to $\sqrt{\left(\frac{l_2}{l_1}\right)}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image moments [13]</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>first Hu moment</td>
</tr>
<tr>
<td>I2</td>
<td>second Hu moment</td>
</tr>
<tr>
<td>pxx</td>
<td>surface in pixels.</td>
</tr>
<tr>
<td>int</td>
<td>the mass of the object.</td>
</tr>
<tr>
<td>n20</td>
<td>translation/scale invariant moments</td>
</tr>
<tr>
<td>n11</td>
<td>translation/scale invariant moments</td>
</tr>
<tr>
<td>n02</td>
<td>translation/scale invariant moments</td>
</tr>
<tr>
<td>l1</td>
<td>major axis</td>
</tr>
<tr>
<td>l2</td>
<td>minor axis</td>
</tr>
<tr>
<td>lk</td>
<td>k-th Hu’s moment.</td>
</tr>
<tr>
<td>theta</td>
<td>orientation</td>
</tr>
<tr>
<td>asm</td>
<td>Angular second moment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Haralick texture features [13]</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>con</td>
<td>Contrast</td>
</tr>
<tr>
<td>cor</td>
<td>Correlation</td>
</tr>
<tr>
<td>var</td>
<td>Variance</td>
</tr>
<tr>
<td>idm</td>
<td>Inverse difference moment</td>
</tr>
<tr>
<td>sav</td>
<td>Sum average</td>
</tr>
<tr>
<td>sva</td>
<td>Sum variance</td>
</tr>
<tr>
<td>sen</td>
<td>Sum entropy</td>
</tr>
<tr>
<td>ent</td>
<td>Entropy</td>
</tr>
<tr>
<td>dva</td>
<td>Difference variance</td>
</tr>
<tr>
<td>den</td>
<td>Difference entropy</td>
</tr>
<tr>
<td>fl2</td>
<td>Measure of correlation</td>
</tr>
<tr>
<td>fl3</td>
<td>Measure of correlation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Zernike moments features based on Zernike polynomial [10]</th>
<th>49</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Spatial feature</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of neighboring cells in a distance of 50</td>
<td></td>
</tr>
</tbody>
</table>