# FCSlib: Fluorescence fluctuation spectroscopy analysis for mobility, number, and molecular brightness in R

*R. Pinto-Cámara, A. Linares-Castañeda, H.O. Hernández, D.S. Moreno-Gutiérrez, J.M. Rendón-Mancha, C.D. Wood & A. Guerrero*

*27/2/2020*

## 1. Step-by-step guide for the use of FCSlib

### 1.1. Data sets

In order to demonstrate the analytical capabilities of the FCSlib package, there is a total of four data sets available, provided as examples of experimental data. One of these data sets ("Cy5_100nM")[1] corresponds to Cy5 molecules freely diffusing in solution and is used to test the Fluorescence Correlation Spectroscopy (FCS) and FitFCS functions. The other three datasets correspond to experimental data in a biological context("V2", "V6" & "V2V6"); they are used to test the Pair Correlation Function (pCF), Number & Brightness (N&B) and Pair Correlation of Molecular Brightness (pCOMB) functions. Information such as abundance, mobility and stoichiometry will be derived through the extensive use of the FCS, pCF, N&B and pCOMB methods. Due to space limitations in CRAN, none of the aforementioned data sets are included in the package, though, they should be downloaded from (https://github.com/RPintoC/FCSlib_Sup_Data).

First, download and install the "FCSlib" package from the CRAN repository.

Type this code on your R script:

```
install.packages("FCSlib")
```

FCSlib handles data by using image manipulation tools. The "tiff" library must be installed for data reading. Type:

```
install.packages("tiff")
```

Once installed, FCSlib should be loaded by typing

```
library(FCSlib)
```

```
## Loading required package: tiff
```

### 1.2. General Correlation Function: assessing the mobility and abundance of free difusing Cy5 molecules

Fluorescence Correlation Spectroscopy (FCS) is a set of non-invasive techniques based on the analysis of fluorescence intensity fluctuations caused by the molecular traffic through the observation volume, where the fluorescence signal $F(t)$ is acquired over time, at frequencies comparable to those of the molecular dynamics of the process of interest. These fluctuations are obtained by subtracting the mean value of the fluorescence signal from the signal at each time point, and contain information about the kinetics of the molecules under study, even at low concentrations (i.e. those comparable to the concentrations of organic molecules in the cytoplasm of live cells). By using single-point analysis, FCS provides information about the dynamics of single molecules within the sample. A signal fluctuation is defined as

$\delta F(t) = F(t) - <F(t)>$ ,

where $F(t)$ is the signal intensity at a given time point and $<F(t)>$ is the mean value of the signal. Then, the correlation between $\delta F(t)$ and $\delta F(t+\tau)$ is calculated using the General Autocorrelation Function:

$G(\tau) = \frac{<\delta F(t)\delta F(t+\tau)>}{<F(t)>^2}$ ,

where $t$ refers to the time point of acquisition, $\tau$ is the delay time at which $\delta F(t)$ is computed and $< ... >$ indicates average. The value of $G(\tau)$ represents the similarity of the signal with itself through different time intervals $\tau$. This means that, for example, if the position of a molecule at a given time $t$ is very similar to its position at a time $t + \tau$, there will be a positive contribution to $G(\tau)$. On the other hand, $G(\tau)$ will decrease as the molecule position changes at longer correlation times.

### 1.2.1. Computing the autocorrelation curves

We will start by analyzing the fluorescence signal of a solution of Cy5 dye collected on a confocal microscope with single photon excitation.

Use the readFileTiff() function to read the fcs data in TIF format.

```
f <- readFileTiff("Cy5_100nM.tif")
```

```
dim(f)
```

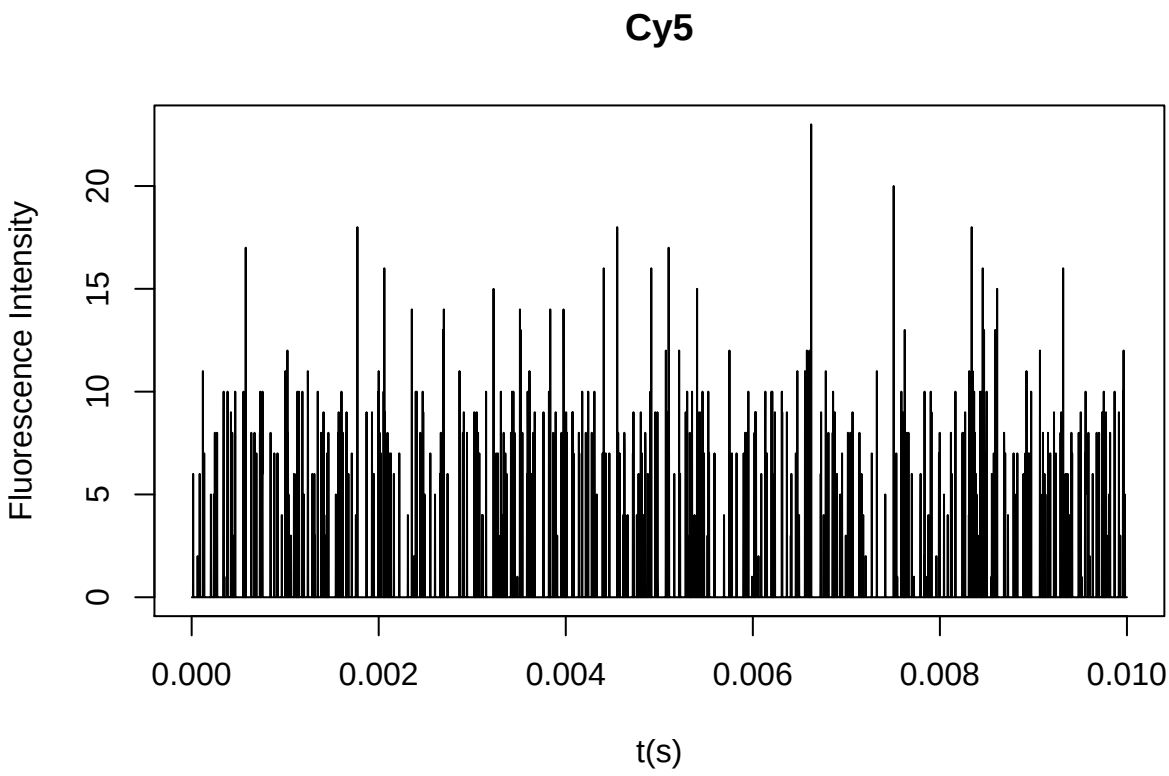
```
## [1] 2048 5000    1
```

Note that $f$ is a matrix of $2048 \times 5000 \times 1$ dimensions. This is due to the fact that this single-point FCS experiment was collected at intervals of 2048 points each, with an acquisition time of 2 $\mu s$. Let's now create a dataframe with the FCS data which here-and-after will be called Cy5.

```
acqTime = 2E-6
f <- as.vector(f)
time <- (1:length(f))*acqTime
Cy5 <- data.frame(t = time, f)
```

The first 10 ms of the time series are:

```
plot(Cy5[1:5000,], type ="l", xlab = "t(s)", ylab ="Fluorescence Intensity", main = "Cy5")
```



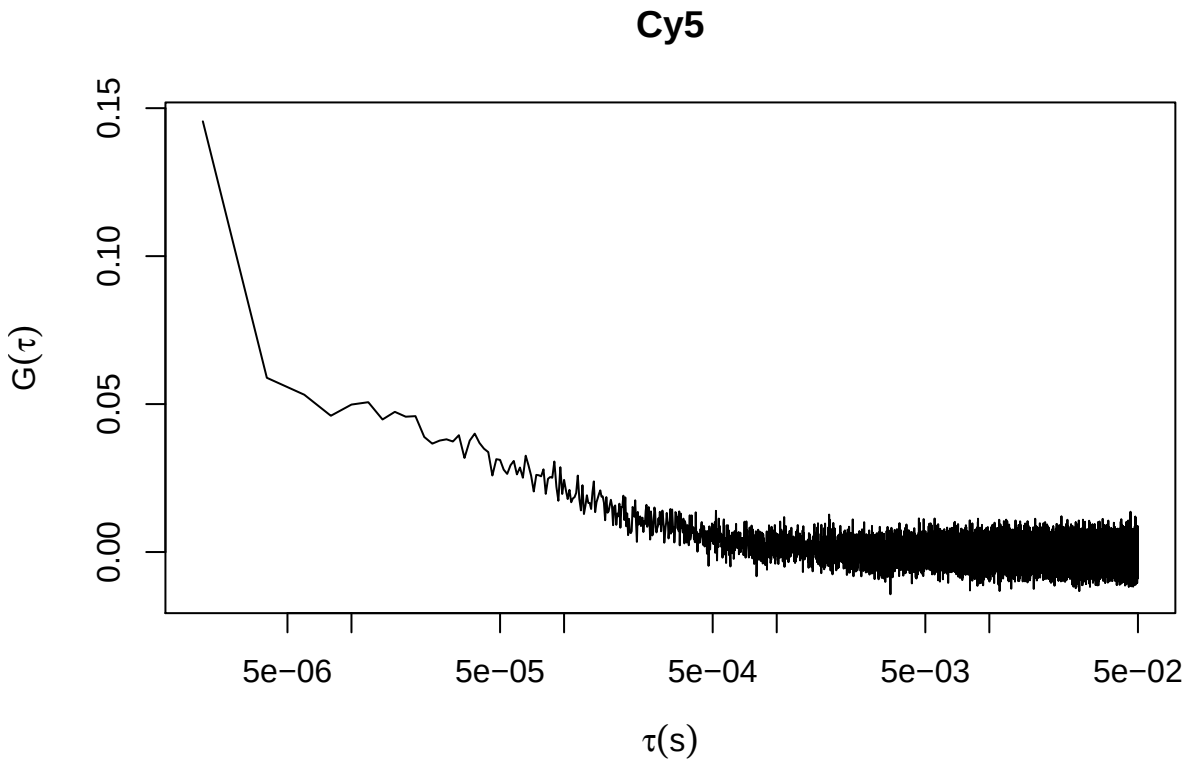

The fcs() function receives three parameters: 'x' (mandatory), 'y'(optional) and 'nPoints' (optional), where x is the main signal to analyze, y is a secondary signal (for the case of cross-correlation instead of autocorrelation) and nPoints is the final length of the calculated correlation curve. This function divides the original N-size signal into sub-vectors with a size of nPoints*2. Once all the sub-vectors are analyzed, these are then averaged.

To use the fcs() function type

```
g <- fcs(x = Cy5$f)
```

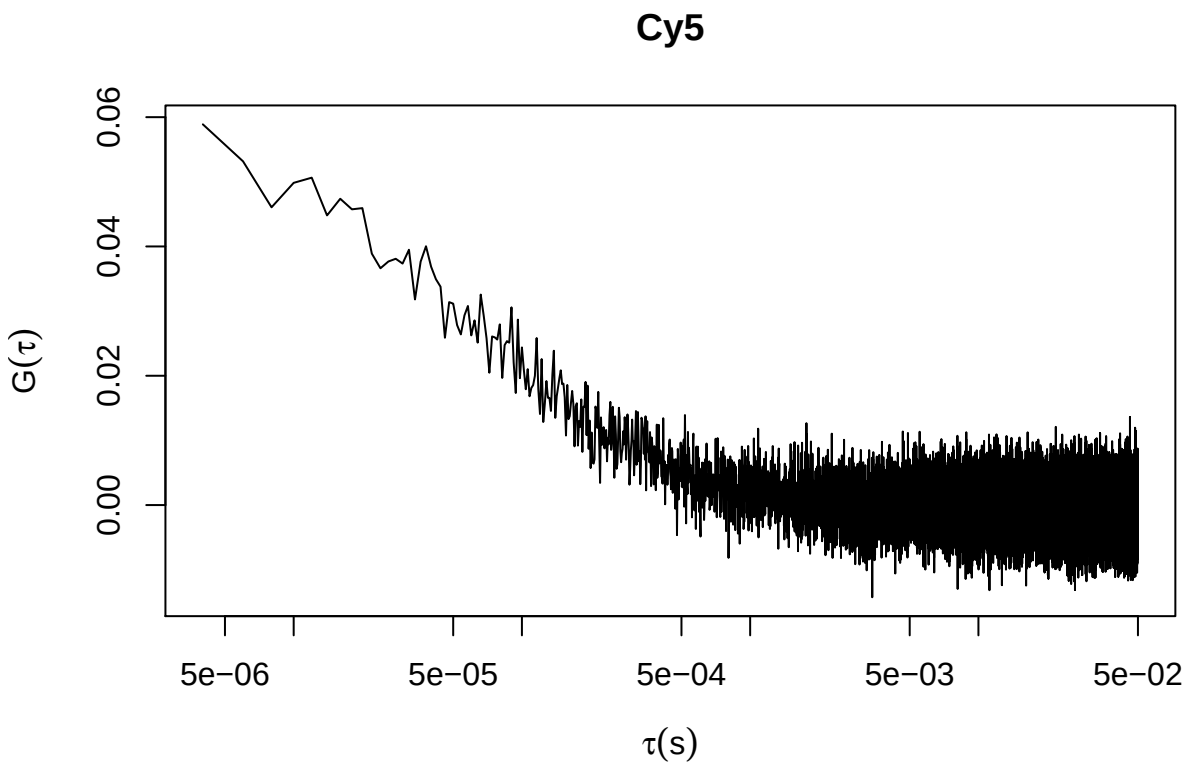
The result of the function has been assigned to the variable 'g', which contains the autocorrelation curve.

```
len <- 1:length(g)
tau <- Cy5$t[len]
G <- data.frame(tau,g)
plot(G, log = "x", type = "l", xlab = expression(tau(s)),
                               ylab = expression(G(tau)), main = "Cy5")
```



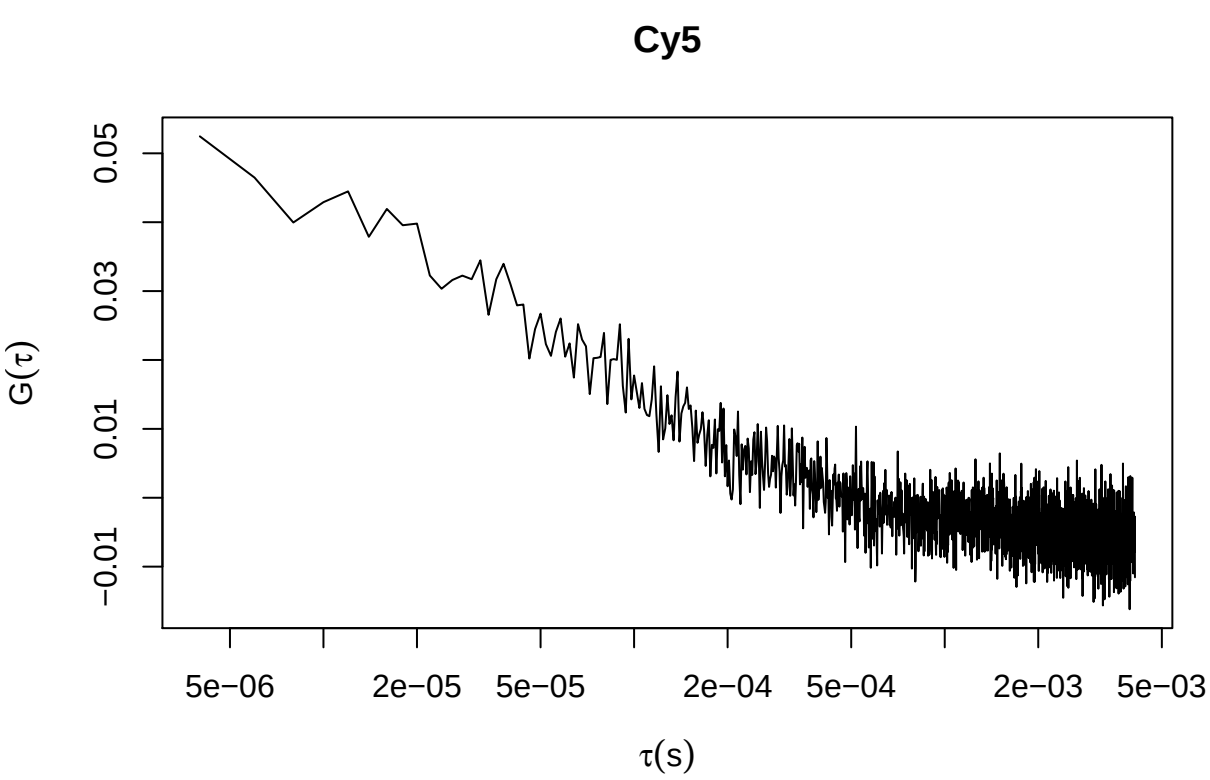

It is important to remove the first point from the data, where $G(\tau = 0)$ is not properly computed.

```
G <- G[-1,]
plot(G, log = "x", type = "l", xlab = expression(tau(s)),
                               ylab = expression(G(tau)), main = "Cy5")
```




The variable 'nPoints' can be adjusted to better assess the transport phenomena in study (i.e. free diffusion in three dimensions, the case of this example) and for better understanding of the diffusive nature of the molecules. In this example 'nPoints' will be set to 2048.

```
g <- fcs(x = Cy5$f,nPoints = 2048)
len <- 1:length(g)
tau <- Cy5$t[len]
G <- data.frame(tau,g)
G <- G[-1,]
plot(G, log = "x", type = "l", xlab = expression(tau(s)),
                               ylab = expression(G(tau)), main = "Cy5")
```




### 1.2.2. Fitting a transport model

A transport model, which contains physical information about the diffusive nature of the fluorophores, can be fitted to the autocorrelation data to obtain parameters such as the diffusion coeficient $D$ and the number of molecules within the observation volume $N$.

The fitFCS() function uses the 'Non-linear Least Squares' function to fit a physical model to the data set. There are four possible models to be fit:

"D2D" for two-dimensional diffusion

$G(\tau) = \frac{\gamma}{N}(1 + \frac{\tau}{\tau_D})^{-1}$

"D2DT" for two-dimensional diffusion with triplet state

$G(\tau) = \frac{\gamma}{N}(1 + \frac{B\exp^{(-\tau/\tau_B)}}{1-B})(1 + \frac{\tau}{\tau_D})^{-1}$

"D3D" for three-dimensional diffusion

$G(\tau) = \frac{\gamma}{N}(1 + \frac{\tau}{\tau_D})^{-1}(1 + \frac{s^2}{u^2}\frac{\tau}{\tau_D})^{-1/2}$

"D3DT" for three-dimensional diffusion with triplet state

$G(\tau) = \frac{\gamma}{N}(1 + \frac{B\exp^{(-\tau/\tau_B)}}{1-B})(1 + \frac{\tau}{\tau_D})^{-1}(1 + \frac{s^2}{u^2}\frac{\tau}{\tau_D})^{-1/2}$

where $\gamma$ is a geometric factor that depends on the illumination profile. For confocal excitation its magnitude approaches to $\gamma = 1/\sqrt{8} \approx 0.35$ fl [1]. The diffusion time is defined as $\tau_D = s^2/4D$, where $s$ and $u$ are the radius and the half-length of the focal volume, respectively. The parameter $u$ is usually expressed as $u = ks$, with $k$ being the eccentricity of the focal volume; for confocal excitation $k \approx 3$ [1]. The fraction of molecules in the triplet state is $B$, and $\tau_B$ is a time constant for the triplet state.

Once the correlation curve 'g' has been generated, a data frame containing known parameters must be then defined. The radius of the focal volume must be determined experimentally and varies for each specific optical system. For this example, we choose a $s = 0.27\ \mu m$, as described in [1]

```
df <- data.frame(G, s = 0.27, k = 3)
head(df)
```

```
##       tau          g    s k
## 2 4.0e-06 0.05246314 0.27 3
## 3 6.0e-06 0.04647721 0.27 3
## 4 8.0e-06 0.03995816 0.27 3
## 5 1.0e-05 0.04290619 0.27 3
## 6 1.2e-05 0.04447412 0.27 3
## 7 1.4e-05 0.03787735 0.27 3
```

Then, three lists that contain the initial values of the data, as well as the upper and lower limits of these values, must be defined. The input values shown here are the expected ones for the real experimental data to be very similar or close to. This step is important for the accurate computation of the real parameters.

Initial values:

```
start <- list(D = 100, G0 = 0.1)
up <- list(D = 1E3, G0 = 10)
low <- list(D = 1E-1, G0 = 1E-2)
```

Once the known parameters are defined, we now proceed to use the fitFCS() function. The result is the fitted model:

```
modelFCS <- fitFCS(df, start, low, up, type = "D3D", trace = F)
summary(modelFCS)
```

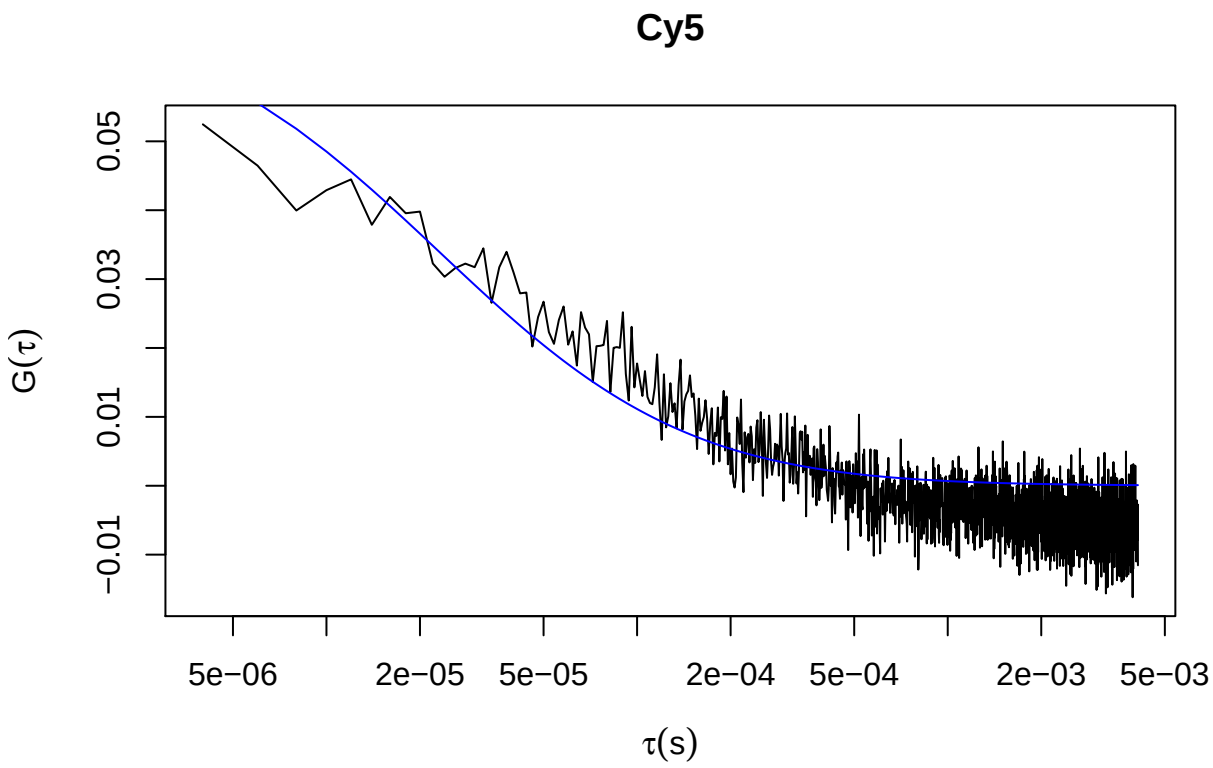
```
##
## Formula: g ~ ((G0) * ((1 + ((4 * D * tau)/(s^2)))^(-1)) * ((1 + ((4 *
##     D * tau)/((k^2) * (s^2))))^(-1/2)))
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## D  7.709e+02  7.235e+01   10.66   <2e-16 ***
## G0 7.064e-02  4.279e-03   16.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005779 on 2045 degrees of freedom
##
## Algorithm "port", convergence message: relative convergence (4)
```

By using the predict() function, the fitted model object generated in the previous step is transformed into a vector that contains the curve fitted by the desired model.

```
fit <- predict(modelFCS, tau)
```

Finally, use the following command to obtain the resulting graph, where the blue line corresponds to the fitted data and the black corresponds to the unfitted.

```
plot (G, log = "x", type = "l", xlab = expression(tau(s)),
                                ylab = expression(G(tau)), main = "Cy5")
lines(fit~G$tau, col = "blue")
```




To access to the physical coefficients of the model type

```
s <- summary(modelFCS)
s$coefficients[,1]
```

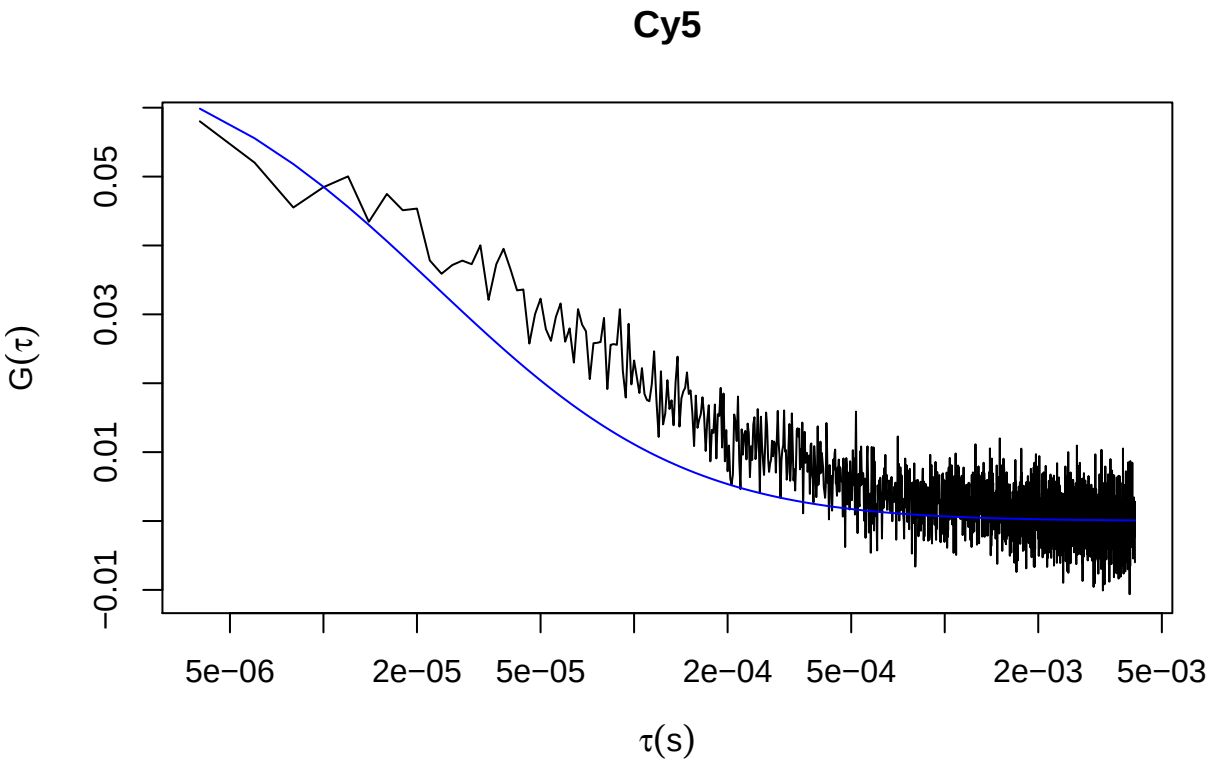
```
##            D           G0
## 770.93426104   0.07064433
```

Note that this fitting yields a $D = 771\ \mu m * s^{-1}$, which is bigger than the reported value for Cy5 in solution, $D = 280\ \mu m * s^{-1}$ [1]. The latter can be explained by the poor fitting obtained, as our $G(\tau)$ data contains negative values. To compensate for this issue, we will estimate the mean value of $G(\tau \to \infty)$, and add it to the $G(\tau)$ data.

```
m <- abs(mean(tail(G$g, n = 100)))
G$g <- G$g + m
modelFCS <- fitFCS(df, start, low, up, type = "D3D", trace = F)
s <- summary(modelFCS)
s$coefficients[,1]
```

```
##            D           G0
## 770.93426104   0.07064433
```

```
fit <- predict(modelFCS, tau)
plot (G, log = "x", type = "l", xlab = expression(tau(s)),
                                ylab = expression(G(tau)), main = "Cy5")
lines(fit~G$tau, col = "blue")
```

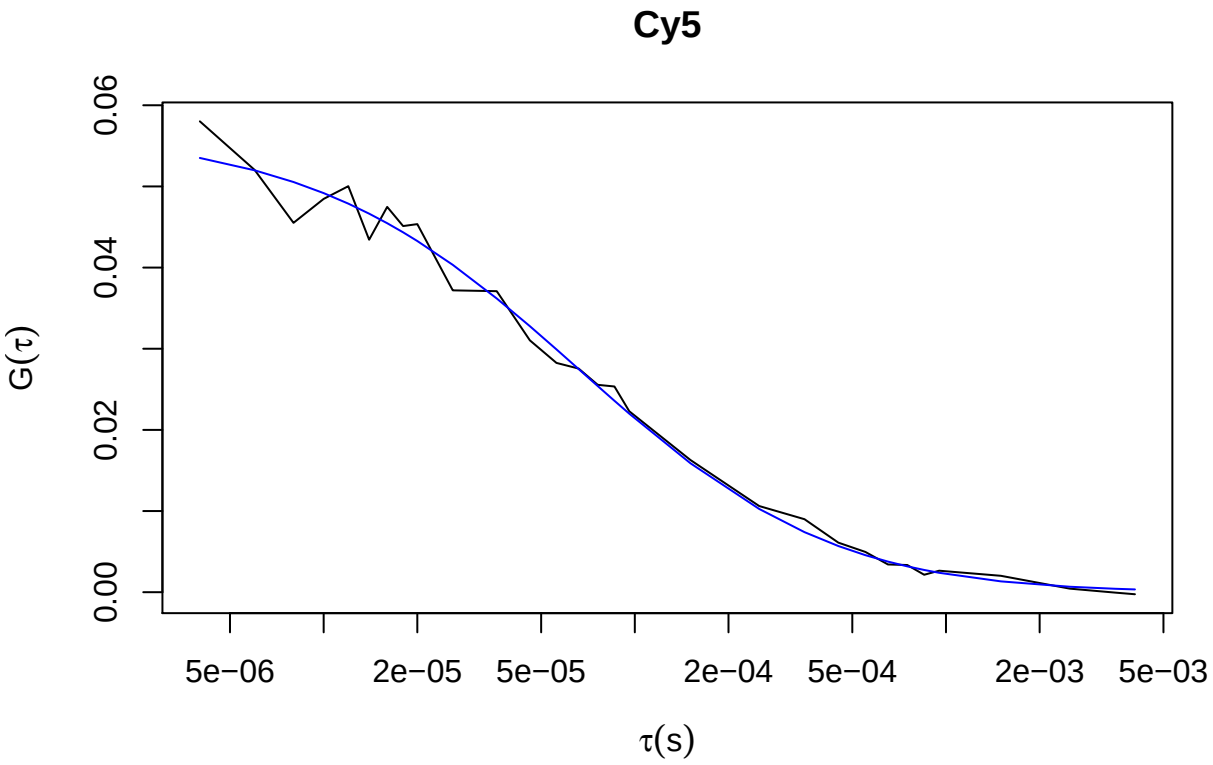



To prevent overweighting of the model in the region of zero correlation, it is useful to average $G$ at longer $\tau$ values

```
sfcs <- simplifyFCS(G$g, G$tau)
df <- data.frame(sfcs, s = 0.27, k = 3)
modelFCS <- fitFCS(df, start, low, up, type = "D3D", trace = F)
summary(modelFCS)
```

```
##
## Formula: g ~ ((G0) * ((1 + ((4 * D * tau)/(s^2)))^(-1)) * ((1 + ((4 *
##     D * tau)/((k^2) * (s^2))))^(-1/2)))
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## D  2.663e+02  1.615e+01   16.49 5.95e-16 ***
## G0 5.682e-02  1.026e-03   55.37  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001838 on 28 degrees of freedom
##
## Algorithm "port", convergence message: relative convergence (4)
```

```
fit <- predict(modelFCS, df$tau)
plot (df$g~df$tau, log = "x", type = "l", xlab = expression(tau(s)),
                                          ylab = expression(G(tau)), main = "Cy5")
lines(fit~df$tau, col = "blue")
```




Now, a $D = 266\ \mu m s^{-1}$ is obtained, which is closer to the reported value for Cy5 dye in solution. The average number of molecules within the observation volume is $N = \gamma/G_0 \approx 6.5$, and the concentration of molecules can be computed as $C = N/V_{eff}$, where $V_{eff} = \pi^{3/2} s^2 u \approx 0.33\ fL$ (for single photon excitation). Hence, the concentration of Cy5 in this experiment should be close to 33 $nM$.

## 1.3. Assessing abundance, mobility and stoichiometry of Venus fluorescent protein oligomers in live cells

In this section, we will analyze the dynamics of dimers and hexamer of Venus, an enhanced yellow fluorescent protein [2], and their ability to cross the nuclear membrane in live cells.

The maximum diameter of molecules capable of crossing the nuclear pore complex in mammals through active transport is ~38 nm, and ~10 nm for passive transport in the case of small molecules [3]. Nevertheless, only those proteins with a diameter smaller or equal to ~5 nm can freely diffuse through the nuclear pore complex [4]. Even though the exact hydrodynamic radius of the Venus dimers and hexamers remain unknown, we predicted the Venus oligomers approximate dimensions by following the relation $radius \propto \sqrt[3]{volume} \propto \sqrt[3]{mass}$. Assuming that the hydrodynamic radius of Venus is about 2.8 nm, the stokes radius of GFP [5], the resulting radius of the dimeric and hexameric species will approach to 3.5 nm and 5.1 nm, respectively. Hence, the hypothesis to test is that the dimers will be able to access the nucleus, while the hexamers will not.

### 1.3.1. Assessing abundance of Venus oligomers at the cellular millieu

As further described in section 2 of this document, fluorescence data acquisition was performed through a raster line scan. The linear region of the scan was 3.2 $\mu m$ long (64 pixels of 0.05 $\mu m$), with a scanning direction from cytoplasm to nucleus. All experiments were acquired with a pixel dwell time of12.5 $\mu s$, and line scan time of 1.925$ms$. The data sets needed for this section should be downloaded (https://github.com/RPintoC/FCSlib_Sup_Data).

In order to ensure valid analysis, we will only consider the first 25,000 points of the time series. This is due to possible sample movement that can usually occur after several seconds of live cell imaging:

```
    v2 <- readFileTiff("V2.tif")[,,1]
    v2 <- v2[,1:25000]
    v6 <- readFileTiff("V6.tif")[,,1]
    v6 <- v6[,1:25000]
    v2v6 <- readFileTiff("V2V6.tif")[,,1]
    v2v6 <- v2v6[,1:25000]
```

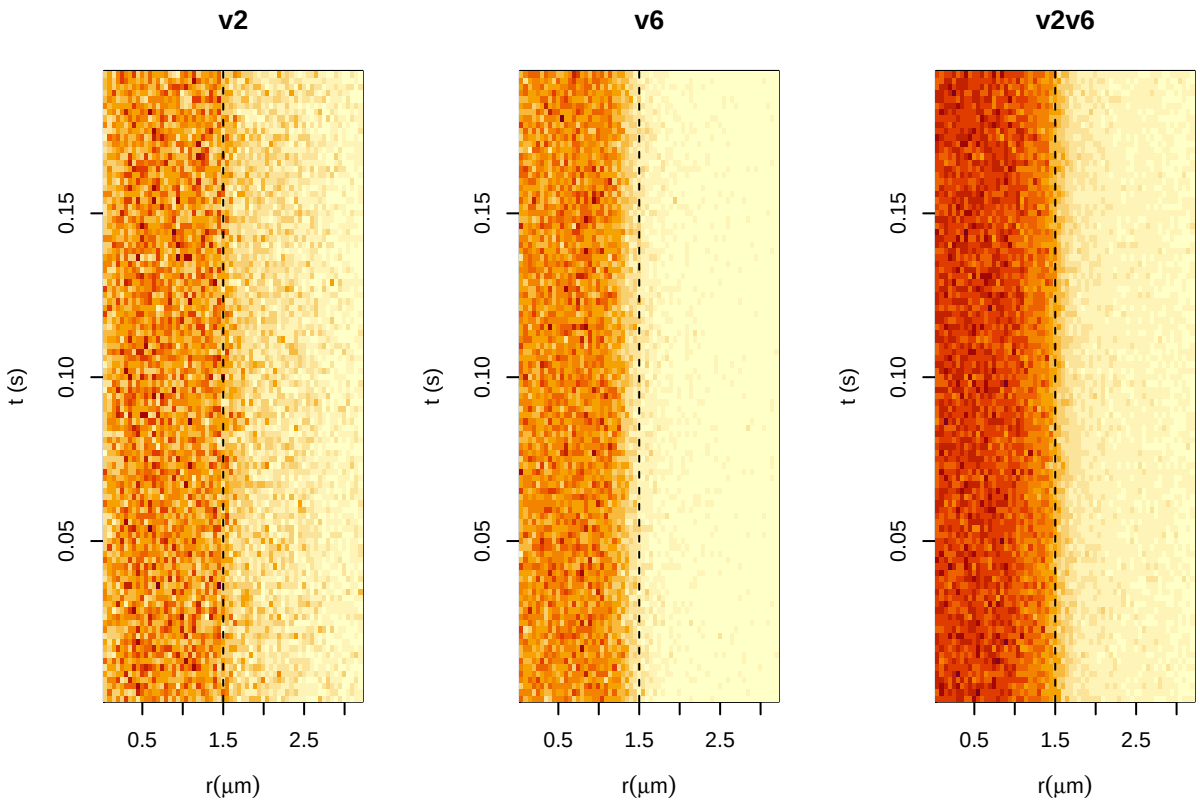
All data sets must be rearranged into 2D matrices (intensity carpets) for proper extraction and usage. An intensity carpet is a graphic spatio-temporal representation of fluorescence signal intensity and/or processed fluorescence correlation data.

To define the temporal "Time" and spatial "r" ranges of these experiments, type:

```
lineTime = 0.001925
pixelSize = 0.05
Time <- (1:dim(v2)[2])*lineTime
r <- (1:dim(v2)[1])*pixelSize
```

To visualize the first 100 time points, type:

```
n <- 100
par(mfrow = c(1,3))
image(y = Time[1:n], x = r, z=v2[, 1:n], xlab=expression(r(mu*m)),
                                         ylab="t (s)", main = "v2")
abline(v = 1.5, lty = 2)
image(y = Time[1:n], x = r, z=v6[, 1:n], xlab=expression(r(mu*m)),
                                         ylab="t (s)", main = "v6")
abline(v = 1.5, lty = 2)
image(y = Time[1:n], x = r, z=v2v6[, 1:n], xlab=expression(r(mu*m)),
                                           ylab="t (s)", main = "v2v6")
abline(v = 1.5, lty = 2)
```




The left portion of each carpet corresponds to the cytoplasm and the right one to the nucleus, while the mid region corresponds to the nuclear envelope (pixels 30-34, vertical line at $\approx 1.5\ \mu m$). The color scale of the carpets indicate the magnitude of the fluorescense intensity signal. Red and light yellow colors indicate the highest and the lowest fluorescence inteisity values, respectively. Note that, in any case, the fluorescence is higher in the cytoplasm than in the nuclear milieu.

The function image.plot() of the package "fields" can be used instead the function image(); the former provides a color code bar that is useful for cuantitative comparisions.

```
install.packages("fields")
library(fields)
```

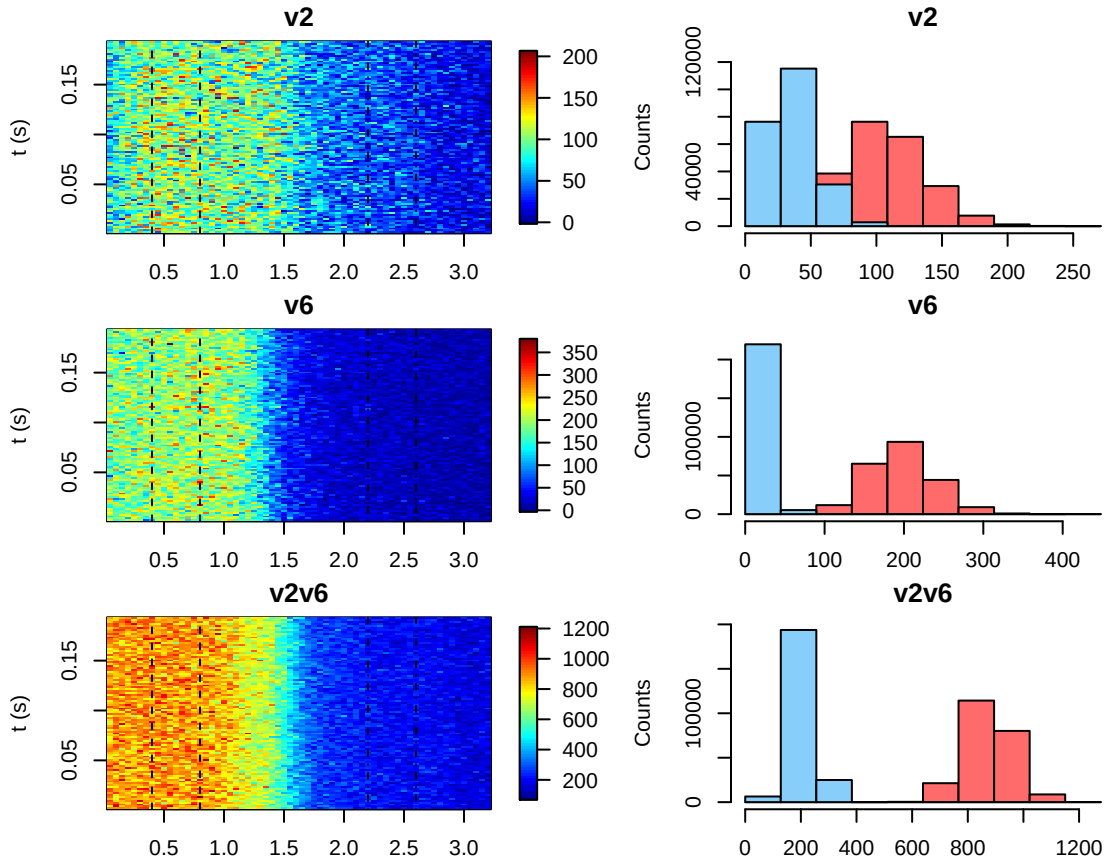
In this representation, the color scale of the carpet directly shows the pixel value. The colder (blue) pixels mean low or near-to-zero signals, and warmer (red) ones are set to the highest signals acquired. Said color scale is not the same for all carpets, though. Some pixels might be the same color as the pixels in another carpet, and it is the color code bar what indicates the actual numerical values.

The histograms of fluorescence are also shown, for a region of 9 pixels ($0.45\ \mu m$) of the raster scan at each, the cytosolic (red bars, pixels 8:16), or at the nuclear compartments (blue bars, pixels 44:52), also indicated as dashed lines at the corresponding fluorescence carpets.

```
par(mfrow = c(3,2), mai = c(5,5,5,5), mar = c(2,2,2,2), pin = c(2,1))
image.plot(y = Time[1:n], x = r, z=v2[, 1:n], xlab=expression(r(mu*m)),
                                                ylab="t (s)", main = "v2")
abline(v = c(0.40, 0.8, 2.2, 2.6), lty = c(2,2,4,4))
breaks <- seq(0, max(v2), max(v2)/10)
hist(v2[8:16,], main="v2", xlab="fluorescence", freq = T, ylim = c(0, 1.3E5),
     xlim=c(0,max(v2)), ylab="Counts", breaks=breaks, col="indianred1")
hist(v2[44:52,], breaks=breaks, col="lightskyblue", freq = T, add=T)
```

```
image.plot(y = Time[1:n], x = r, z=v6[, 1:n], xlab=expression(r(mu*m)),
                                                ylab="t (s)", main = "v6")
abline(v = c(0.40, 0.8, 2.2, 2.6), lty = c(2,2,4,4))
breaks <- seq(0, max(v6), max(v6)/10)
hist(v6[8:16,], main="v6", xlab="fluorescence", freq = T, ylim = c(0, 2.3E5),
     xlim=c(0,max(v6)), ylab="Counts", breaks=breaks, col="indianred1")
hist(v6[44:52,], breaks=breaks, col="lightskyblue", freq = T, add=T)

image.plot(y = Time[1:n], x = r, z=v2v6[, 1:n], xlab=expression(r(mu*m)),
                                                  ylab="t (s)", main = "v2v6")
abline(v = c(0.40, 0.8, 2.2, 2.6), lty = c(2,2,4,4))
breaks <- seq(0, max(v2v6), max(v2v6)/10)
hist(v2v6[8:16,], main="v2v6", xlab="fluorescence", freq = T, ylim = c(0, 2E5),
     xlim=c(0,max(v2v6)), ylab="Counts", breaks=breaks, col="indianred1")
hist(v2v6[44:52,], breaks=breaks, col="lightskyblue", freq = T, add=T)
```




Note that the total fluorescence signal in the v2 carpet is approximately one third of that in the v6 carpet, probably due to a v6/v2 stoichiometry ratio of about 3.

The fluorescence in the nucleus of the cell expressing dimers (v2) is about one third of that in the cytosol (left pannels), which probably means that, although they can transit through the nuclear pore complex, their abundance is distinct and so is the partition of v2 proteins within such subcellular microenviroments.

The fluorophore concentration in the nucleus of cells expressing sole hexamers (v6) appear to be very low (less than 100 digital levels of fluorescence and near-to-zero values) (middle panels). This may indicate that, while Venus dimers are able to transit through the nuclear pore complex, the hexamers cannot.

As expected, cells expressing both v2 and v6 oligomers (v2v6) show higher levels of fluorescence which, in any case, were more abundant in the cytosol (right panel).

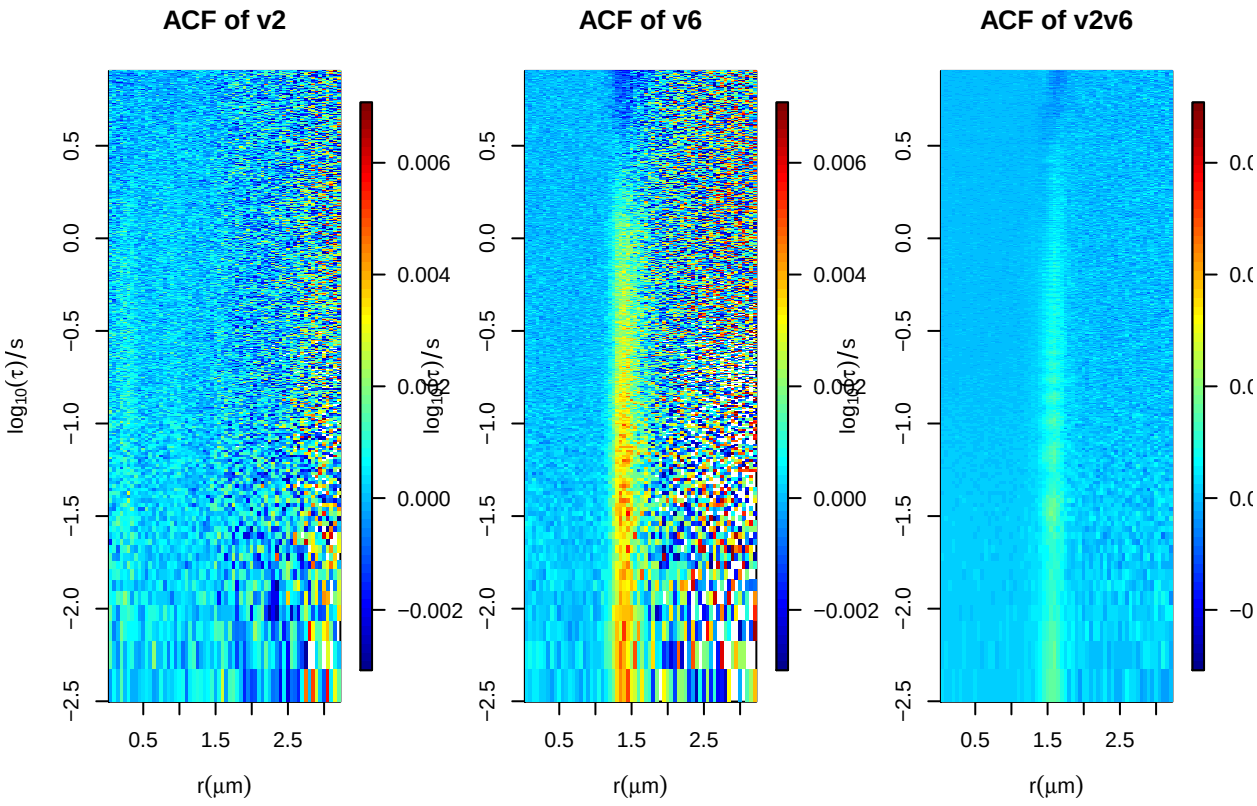
### 1.3.2. Assessing the mobility of Venus dimers and hexamers at the cytoplasm, the nucleus and across the nuclear envelope

The aim of this section is to address the mobility of v2 and v6 within the cytoplasm, inside the nucleus and across the nuclear membrane. We implemented the fcs() function, but at single pixel ($r$) all along the raster scan line

```
nPoints <- dim(v2)[2]/6
v2.acf <- v6.acf <- v2v6.acf <- NULL
for(px in 1:dim(v2)[1]){
  v2.acf <- rbind(v2.acf,fcs(v2[px,], nPoints =nPoints))
  v6.acf <- rbind(v6.acf,fcs(v6[px,], nPoints =nPoints))
  v2v6.acf <- rbind(v2v6.acf,fcs(v2v6[px,], nPoints =nPoints))
}
v2.acf <- v2.acf[,-1]
v6.acf <- v6.acf[,-1]
v2v6.acf <- v2v6.acf[,-1]
tau <- (2:nPoints)*(lineTime)
```

```
par(mfrow = c(1,3))

image.plot(y = log10(tau), x = r, z=v2.acf, zlim = c(-0.003, 0.007),
           xlab=expression(r(mu*m)), ylab= expression(log[10](tau)/s),
           main = "ACF of v2")
image.plot(y = log10(tau), x = r, z=v6.acf,zlim = c(-0.003, 0.007),
           xlab=expression(r(mu*m)), ylab= expression(log[10](tau)/s),
           main = "ACF of v6")
image.plot(y = log10(tau), x = r, z=v2v6.acf,zlim = c(-0.003, 0.007),
           xlab=expression(r(mu*m)), ylab= expression(log[10](tau)/s),
           main = "ACF of v2v6")
```




The $x$ axis of these temporal-autocorrelation carpets indicates the position at the raster scan ($r$), from the cytoplasm to the nucleus, and the $y$ axis indicates the temporal autocorrelation $G(\tau)$ computed at each pixel $r$.

To simplify the temporal autocorrelation carpets type:

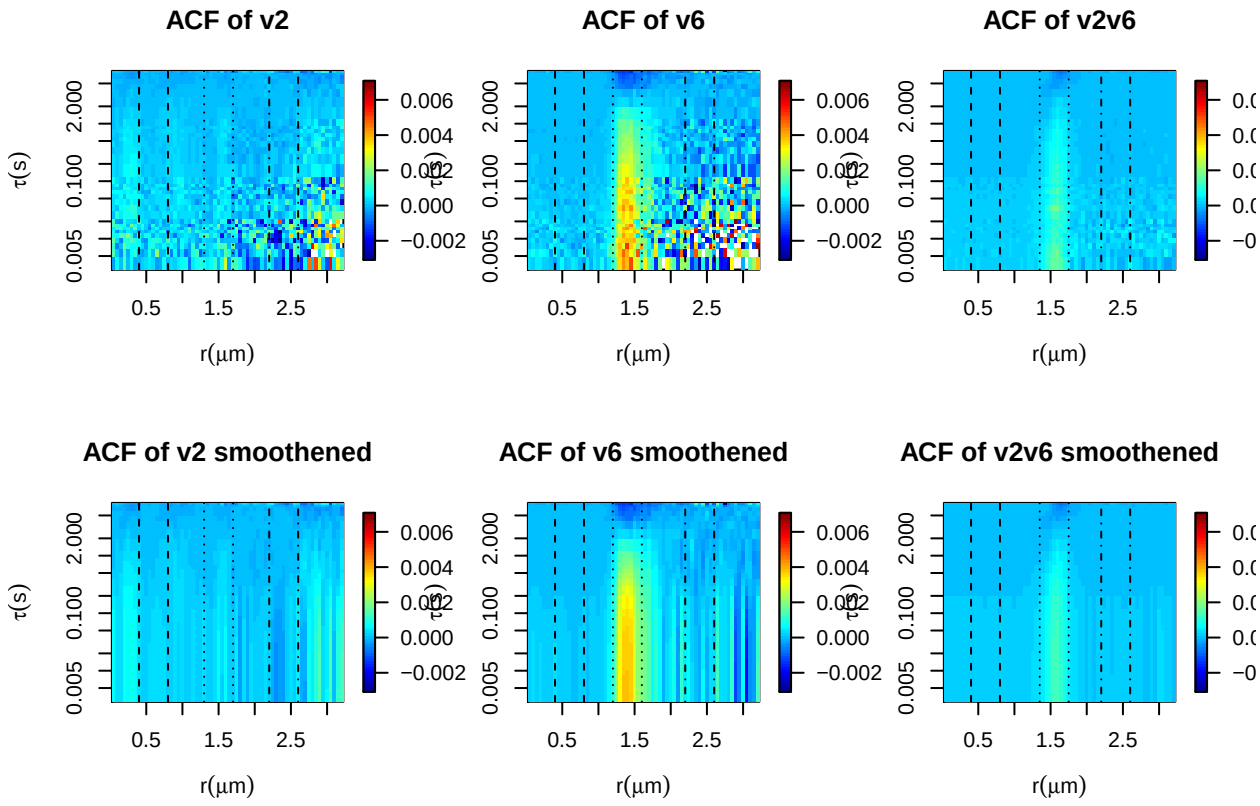
```
v2.g <- v6.g<-v2v6.g<-NULL
v2.gs <- v6.gs<-v2v6.gs<-NULL
for(px in 1:dim(v2)[1]){
  G <- simplifyFCS( v2.acf[px,], tau)
  v2.g <- rbind(v2.g, G$g)
  g <- smooth.spline(G$tau, G$g, df =8)
  v2.gs <- rbind(v2.gs, g$y)
  G <- simplifyFCS( v6.acf[px,], tau)
  v6.g <- rbind(v6.g, G$g)
  g <- smooth.spline(G$tau, G$g, df =8)
  v6.gs <- rbind(v6.gs, g$y)
  G <- simplifyFCS( v2v6.acf[px,], tau)
  v2v6.g <- rbind(v2v6.g, G$g)
  g <- smooth.spline(G$tau, G$g, df =8)
  v2v6.gs <- rbind(v2v6.gs, g$y)
}
```

This procedure averages out near-to-zero values of $G$, when $\tau \to \infty$.

Upper pannels correspond to the simplified autocorrelation functions (ACFs), while the lower ones corresponds to the smoothed data.

```
par(mfrow = c(2,3))
image.plot(y = G$tau, x = r, z = v2.g, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v2")
abline(v = c(0.40, 0.8, 1.3, 1.7, 2.2, 2.6), lty = c(2,2,3,3,4,4))
image.plot(y = G$tau, x = r, z = v6.g, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v6")
abline(v = c(0.40, 0.8, 1.2, 1.6, 2.2, 2.6), lty = c(2,2,3,3,4,4))
image.plot(y = G$tau, x = r, z = v2v6.g, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v2v6")
abline(v = c(0.40, 0.8,1.35, 1.75, 2.2, 2.6), lty = c(2,2,3,3,4,4))

image.plot(y = G$tau, x = r, z = v2.gs, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v2 smoothened")
abline(v = c(0.40, 0.8, 1.3, 1.7, 2.2, 2.6), lty = c(2,2,3,3,4,4))
image.plot(y = G$tau, x = r, z = v6.gs, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v6 smoothened")
abline(v = c(0.40, 0.8, 1.2, 1.6, 2.2, 2.6), lty = c(2,2,3,3,4,4))
image.plot(y = G$tau, x = r, z = v2v6.gs, zlim = c(-0.003, 0.007),log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = "ACF of v2v6 smoothened")
abline(v = c(0.40, 0.8,1.35, 1.75, 2.2, 2.6), lty = c(2,2,3,3,4,4))
```



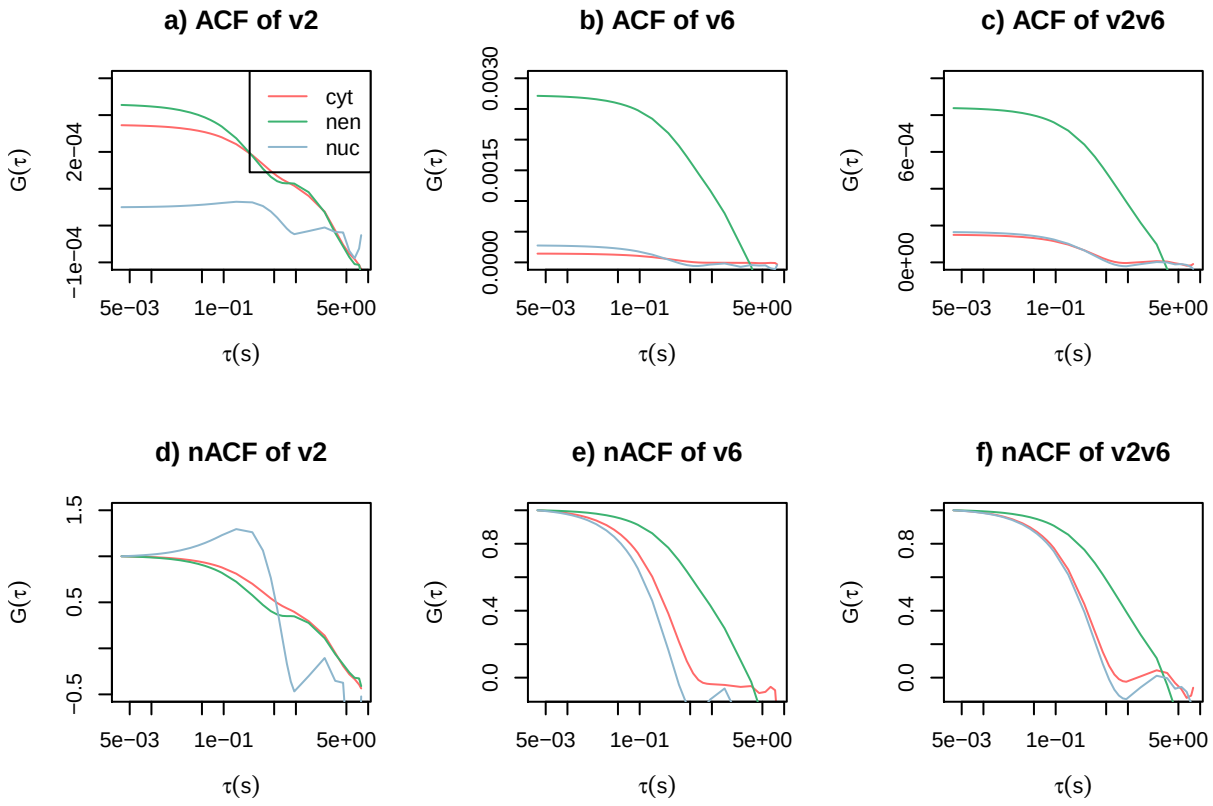

These data show that, there are microenviroments where Venus forms experience distinct mobilities. Dashed lines define three microeviroments, the cytoplasm (cyt), the boundary of the nuclear evelope (nen) and the nucleus (nuc). Note that, regardless the case (v2, v6, or v2/v6), the magnitude of $G(\tau \rightarrow 0)$ is higher at the interface between the cytoplasm and the nucleus. The magnitude of $G(\tau \rightarrow 0)$ is inversely proportional to the number of molecules. The auto-correlation carpets of v2, v6, v2/v6, indicate that the concentration of Venus decreases at the boundary of the nuclear envelope (warmer colors of $G(\tau)$). The latter becomes evident when averaging autocorrelation functions within each subcellular microenviroment (cyt, nen, nuc).

```
v2.g.cyc <- apply(v2.gs[8:16,],MARGIN = 2, mean)
v2.g.nen <- apply(v2.gs[26:34,],MARGIN = 2, mean)
v2.g.nuc <- apply(v2.gs[44:52,],MARGIN = 2, mean)
v6.g.cyc <- apply(v6.gs[8:16,],MARGIN = 2, mean)
v6.g.nen <- apply(v6.gs[24:32,],MARGIN = 2, mean)
v6.g.nuc <- apply(v6.gs[44:52,],MARGIN = 2, mean)
v2v6.g.cyc <- apply(v2v6.gs[8:16,],MARGIN = 2, mean)
v2v6.g.nen <- apply(v2v6.gs[27:35,],MARGIN = 2, mean)
v2v6.g.nuc <- apply(v2v6.gs[44:52,],MARGIN = 2, mean)
```

The following plots correspond to the ACFs computed at distinct subcellular microenviroment: cyt (red lines), nen (green lines), nuc (blue lines) (upper panels). Their corresponding nomalized averaged autocorrelation functions (nACF) are also shown (lower panels).

```
par(mfrow = c(2,3))
plot (v2.g.cyc~G$tau, log = "x", type = "l", xlab = expression(tau(s)),
                          ylim = c(-1E-4,4E-4), ylab = expression(G(tau)),
      main = "a) ACF of v2", col = "indianred1")
lines (v2.g.nen~G$tau, type = "l", col = "mediumseagreen")
```

```
lines (v2.g.nuc~G$tau, type = "l", col =  "lightskyblue3")
legend("topright", legend=c("cyt", "nen", "nuc"),
       col = c("indianred1", "mediumseagreen", "lightskyblue3"), lty = 1)
plot (v6.g.cyc~G$tau, log = "x", type = "l", xlab = expression(tau(s)), ylim = c(0, 3E-3),
      ylab = expression(G(tau)),
      main = "b) ACF of v6", col = "indianred1")
lines (v6.g.nen~G$tau, type = "l", col = "mediumseagreen")
lines (v6.g.nuc~G$tau, type = "l", col =  "lightskyblue3")
plot (v2v6.g.cyc~G$tau, log = "x", type = "l", xlab = expression(tau(s)),
      ylim = c(0, 1E-3), ylab = expression(G(tau)),
      main = "c) ACF of v2v6", col = "indianred1")
lines (v2v6.g.nen~G$tau, type = "l", col = "mediumseagreen")
lines (v2v6.g.nuc~G$tau, type = "l", col =  "lightskyblue3")
plot (v2.g.cyc/mean(v2.g.cyc[1])~G$tau, log = "x", type = "l",
      xlab = expression(tau(s)), ylim = c(-0.5,1.5), ylab = expression(G(tau)),
      main = "d) nACF of v2", col = "indianred1")
lines (v2.g.nen/mean(v2.g.nen[1])~G$tau, type = "l", col = "mediumseagreen")
lines (v2.g.nuc/mean(v2.g.nuc[1])~G$tau, type = "l", col =  "lightskyblue3")
plot (v6.g.cyc/mean(v6.g.cyc[1])~G$tau, log = "x", type = "l",
      xlab = expression(tau(s)), ylim = c(-0.1,1), ylab = expression(G(tau)),
      main = "e) nACF of v6", col = "indianred1")
lines (v6.g.nen/mean(v6.g.nen[1])~G$tau, type = "l", col = "mediumseagreen")
lines (v6.g.nuc/mean(v6.g.nuc[1])~G$tau, type = "l", col =  "lightskyblue3")
plot (v2v6.g.cyc/mean(v2v6.g.cyc[1])~G$tau, log = "x", type = "l",
      xlab = expression(tau(s)), ylim = c(-0.1,1), ylab = expression(G(tau)),
      main = "f) nACF of v2v6", col = "indianred1")
lines (v2v6.g.nen/mean(v2v6.g.nen[1])~G$tau, type = "l", col = "mediumseagreen")
lines (v2v6.g.nuc/mean(v2v6.g.nuc[1])~G$tau, type = "l", col =  "lightskyblue3")
```




The autocorrelation carpet for v2 indicates that, both the concentration and molecular mobility are slightly reduced at the nuclear membrane region. In the case of the v6 carpet, said mobility restriction and concentration decrease is much more evident. Hexamers mobility at the nuclear membrane region is almost completey slowed down, and their concentration is much lower in comparison with the cytoplasm. When expressing sole v2, the mobility of dimers within the nucleus decreased. When expressing sole v6, the mobility of those hexamers that do manage to enter the nucleus, however, remains almost the same as in the cytoplasm. Similarly to the v6 carpet, the v2v6 carpet indicates an even more severe restriction to molecular mobility at the nuclear membrane interface, and the concentration of fluorophores in this area is even lower. Also, note that oligomers in this experiment (specially the hexamers) require longer diffusion times in order to transit through the barrier, as the autocorrelation curves reach higher values in the logarithmic tau scale.

### 1.3.3. The Pair Correlation Function approach: assessing the mobility of single Venus oligomers through the nuclear envelope

This approach analyzes the data of a raster line to extract information about the trajectory of single particles moving across the nuclear envelope, i.e. it measures the time it takes for a particle to move from one point to another. In this method, the temporal cross-correlation function between distinct points in the space (i.e. two different positions across the scan line) is computed. It allows to describe the average path of a certain molecule within the cell structures, creating a map of molecular flux with a spatial resolution provided by the focal length of the imaging volume [6].

$$G(\tau, \delta r) = \frac{<F(t,r) * F(t+\tau, r+\delta r)>}{<F(t,r)> * <F(t,r+\delta r)>} - 1$$

Where $F(t,r)$ is the signal intensity at a given acquisition time $t$ and at a $r$ position across the scan line, $\delta r$ represents the spatial interval (i.e. distance between pixels) at which the cross-correlation is calculated. Here, we introduce the pcf() function, which compares the fluorescence intensity of two points separated by a given time and position, correlating their values and creating a molecular flux map of single molecules.

Both the ACF and pCF methods use the same mathematical principle. In the case of ACF, each pixel is correlated with itself, separated only by a certain time window, which means that 'dr' (distance in pixels) equals zero. On the other hand, for pCF 'dr' is different and greater than zero, as it cross-correlates pixels separated by time and distance.

As a starting approach, we will use pcf() to compare the mobility of v2 and v6 in the cytoplasm of a cell expressing both oligomers, as they cross the nuclear pore complex. The data for pCF is represented as a three-dimensional matrix of position, time and intensity.

When performing pCF, choosing different values for 'dr' can provide the user with different information, depending on the nature of the phenomena in study and, more specifically, the size of the barriers/obstacles to diffusion. In this example, a 'dr' value of 5 pixels (250 nm) is used, since these molecules cannot travel through the nuclear pore complex, they should correlate with themselves at this spatial range.

The pcf() function also allows to calculate the Autocorrelation Function (ACF) for each pixel, since both analysis use the same equation to perform the calculations, with the only difference that in ACF, $\delta r = 0$. To do this, the 'type' parameter must be set to "a".

```
 v2.acf <- pcf(img = v2, nPoints, type = "a")
 v6.acf <- pcf(img = v6, nPoints, type = "a")
 v2v6.acf <- pcf(img = v2v6, nPoints, type = "a")
....
```

To calculate the cross-correlation between spatial domains along the raster scan, which are separated by 0.2 $\mu m$ (dr = 4 pixels), type:

```
dr <- 5
nPoints<-dim(v2)[2]/2
pCv2 <- pcf(img = v2, nPoints, type = "d", dr)
pCv6 <- pcf(img = v6, nPoints, type = "d", dr)
pCv2v6 <- pcf(img = v2v6, nPoints, type = "d", dr)
```

Note that the parameters dr = , and type = "d" are provided to the pcf function. In these examples, the paired-correlated data is organized as a matrix of 64 - dr = 59 columns and nPoints = 5000 rows. Columns (r) represent the physical space scanned in the cell where correlation is computed against a consecutive space $r + \delta r$. Rows represent the correlation time $\tau$. The pseudocolor scale shows the magnitude of the pair correlation fuction $G(\tau, \delta r)$.

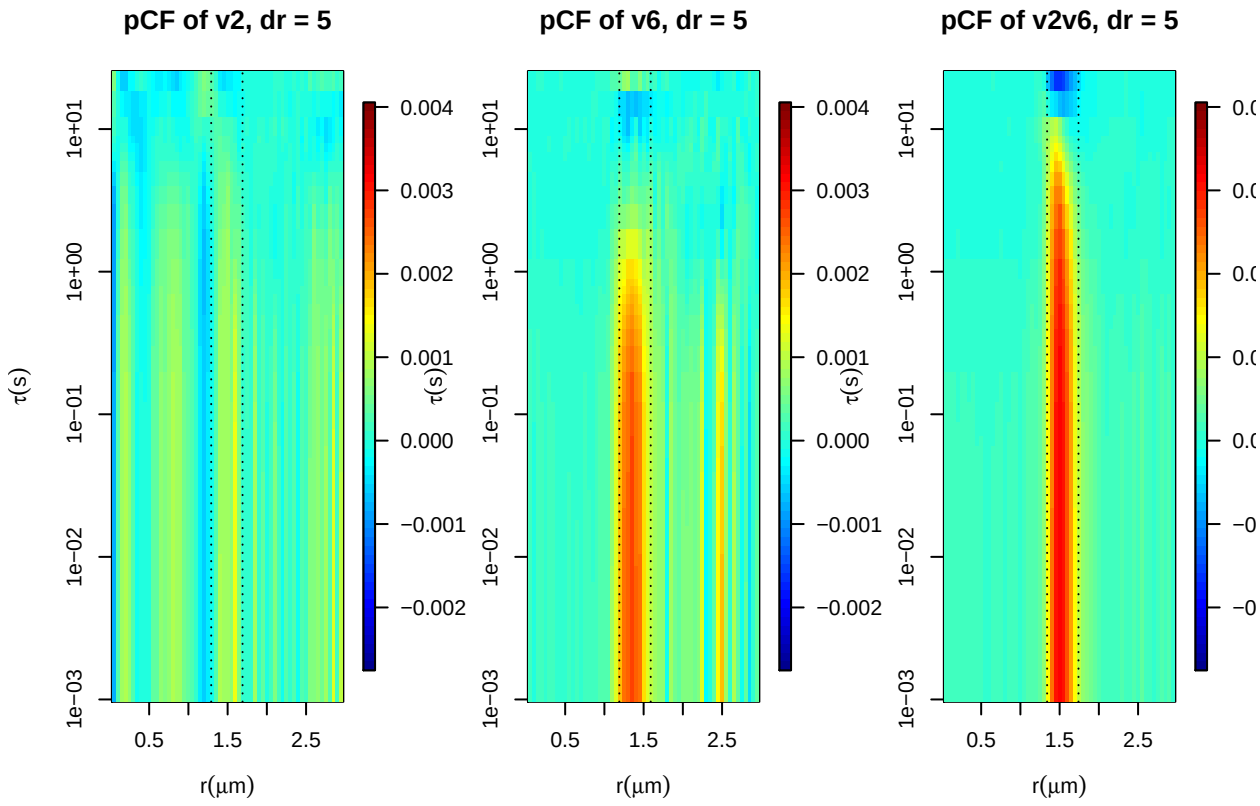
Temporal variables, such as the line scan time and $\tau$ (time window between acquisitions), need to be first defined before data simpliyfing and plot generation.

```
tau <- (1:nPoints)*(lineTime)
r <- (1:dim(pCv2)[1])*pixelSize
```

Simplify and smooth plotted data.

```
  pCv2.gs <- pCv6.gs <- pCv2v6.gs <- NULL
  for(px in 1:dim(pCv2)[1]){
    G <- simplifyFCS(pCv2[px,], tau)        #Simplify v2
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v2
    pCv2.gs <- rbind(pCv2.gs, g$y)
    G <- simplifyFCS(pCv6[px,], tau)        #Simplify v6
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v6
    pCv6.gs <- rbind(pCv6.gs, g$y)
    G <- simplifyFCS(pCv2v6[px,], tau)      #Simplify v2v6
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v2v6
    pCv2v6.gs <- rbind(pCv2v6.gs, g$y)
  }
  tau.s <- g$x
```

```
par(mfrow = c(1,3))
image.plot(y =  tau.s, x = r, z = pCv2.gs,log = "y", zlim = c(-2.7E-3, 4E-3),
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("pCF of v2, dr =",dr))
abline(v = c( 1.3, 1.7) - lineTime*dr, lty = 3)
image.plot(y =  tau.s, x = r, z =pCv6.gs,log = "y", zlim = c(-2.7E-3, 4E-3),
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("pCF of v6, dr =",dr))
abline(v = c(1.2, 1.6) - lineTime*dr, lty = 3)
image.plot(y =  tau.s, x = r, z = pCv2v6.gs,log = "y", zlim = c(-2.7E-3, 4E-3),
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("pCF of v2v6, dr =",dr))
abline(v = c(1.35, 1.75) - lineTime*dr, lty = 3)
```




Notice the prescence of high values of $G(\tau, \delta r = 4)$ at the boundaries of the nuclear envelope, showing a reduced mobility for both v2 and v6 forms. Such a high correlation region observed in the pCF data is an indicative of a barrier to diffusion.

To analyze the directional mobility of venus forms in a region between the cytoplasm and the nuclear envelope type

```
Range = 11:34
pCv2.dat <- list(y = tau.s, x = r[Range],  z = pCv2.gs[Range,])
pCv6.dat <- list(y = tau.s, x = r[Range-2], z =pCv6.gs[Range-2,])
pCv2v6.dat <- list(y = tau.s, x = r[Range+1], z = pCv2v6.gs[Range+1,])
```

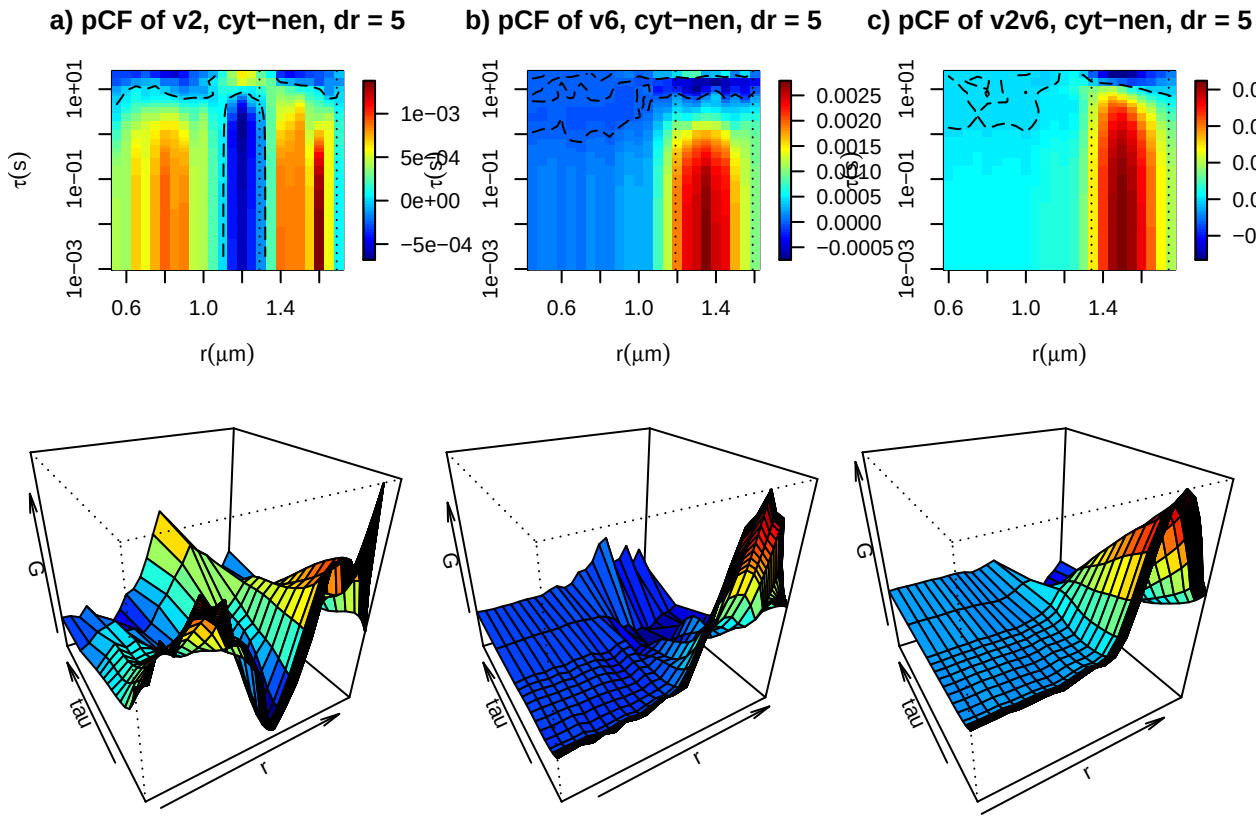
```
compPerpsZval <- function(z, nbcol = 64){
  color <- tim.colors(nbcol)
  nrz <- nrow(z)
  ncz <- ncol(z)
  # Compute the z-value at the facet centres
  zfacet <- z[-1, -1] + z[-1, -ncz] + z[-nrz, -1] + z[-nrz, -ncz]
  facetcol <- cut(zfacet, nbcol)
  return(color[facetcol])
}
######################
par(mfrow = c(2,3))

image.plot(pCv2.dat,log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("a) pCF of v2, cyt-nen, dr =",dr))
contour(pCv2.dat, nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
abline(v = c( 1.3, 1.7) - lineTime*dr, lty = 3)

image.plot(pCv6.dat,log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("b) pCF of v6, cyt-nen, dr =",dr))
contour(pCv6.dat,nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
abline(v = c(1.2, 1.6) - lineTime*dr, lty = 3)

image.plot(pCv2v6.dat,log = "y",
           xlab=expression(r(mu*m)), ylab= expression(tau(s)),
           main = paste("c) pCF of v2v6, cyt-nen, dr =",dr))
contour(pCv2v6.dat,nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
abline(v = c(1.35, 1.75) - lineTime*dr, lty = 3)

par(pin = c(2.1,2.1))
persp(pCv2.dat, col = compPerpsZval(pCv2.dat$z),
      xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
persp(pCv6.dat, col = compPerpsZval(pCv6.dat$z),
      xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
persp(pCv2v6.dat, col = compPerpsZval(pCv2v6.dat$z),
      xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
```




The carpets of v2, v6 and v2v6 contain a level plot showing the transition from negative to positive correlation (dashed lines). These plots show that regardless the Venus form, there is a region of reduced mobility close to the nuclear envelope (delimited by horizontal dashed lines). Interestingly, the carpet of v2 shows a region containing an arch of positive correlation (a, green to yellow) prior to the nuclear envelope, in a range within $r : [0.95, 1.15]\ \mu m$.

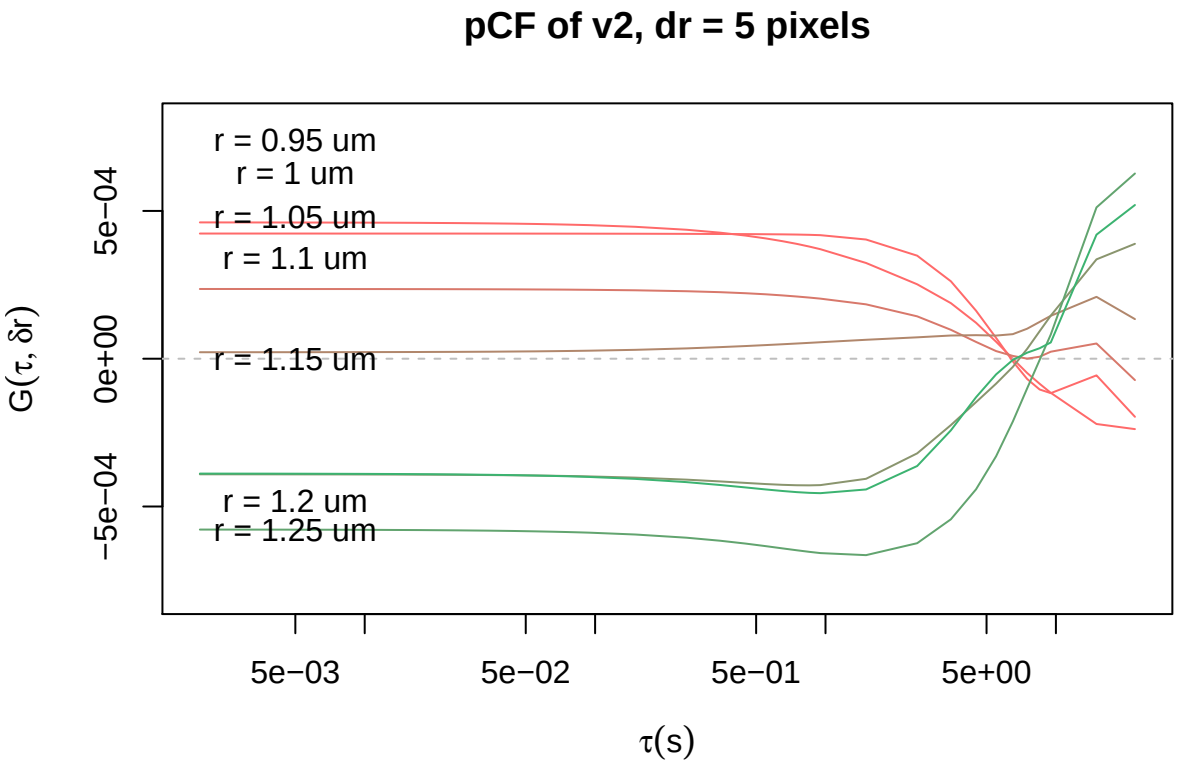
A particle observed at a given time $t$ and location $r$ can be found at a distance $r + \delta r$ with a probability that is proportional to the fluorescence intensity at a given distance. Only the same particle can produce, in average, positive cross-correlation with a give delay time, at two different points: $r$ and $r + \delta r$. Such principle explains the arc of positive cross-correlation observed in the v2 carpet.

The following plots show the pCF carpets computed within the range of $r : [0.95, 1.15]\ \mu m$.

```
layout(1)
pal <- colorRampPalette(c("indianred1", "mediumseagreen"))
Range <- 20:25
COLS <- pal(length(Range+1))
plot (pCv2.gs[19,]~G$tau, log ="x", type = "l",
      xlab = expression(tau(s)),
      ylab = expression(G(tau,delta*r)),
      col = COLS[1],
      ylim = c(-8E-4,8E-4 ),
      main = paste("pCF of v2, dr =",dr, "pixels"))
count <- 1
abline (h = 0, col = "gray", lty = 2)
for ( i in Range){
  lines(pCv2.gs[i,]~G$tau, col = COLS[count])
  count = count + 1
```

```
}

text(x = 5E-3, 7.4E-4,
     paste("r =", 19*pixelSize, "um"))
text(x = 5E-3, c(6.3E-4, 4.8E-4, 3.4E-4, 0,-4.8E-4, -5.8E-4),
     paste("r =", Range*pixelSize, "um"))
```



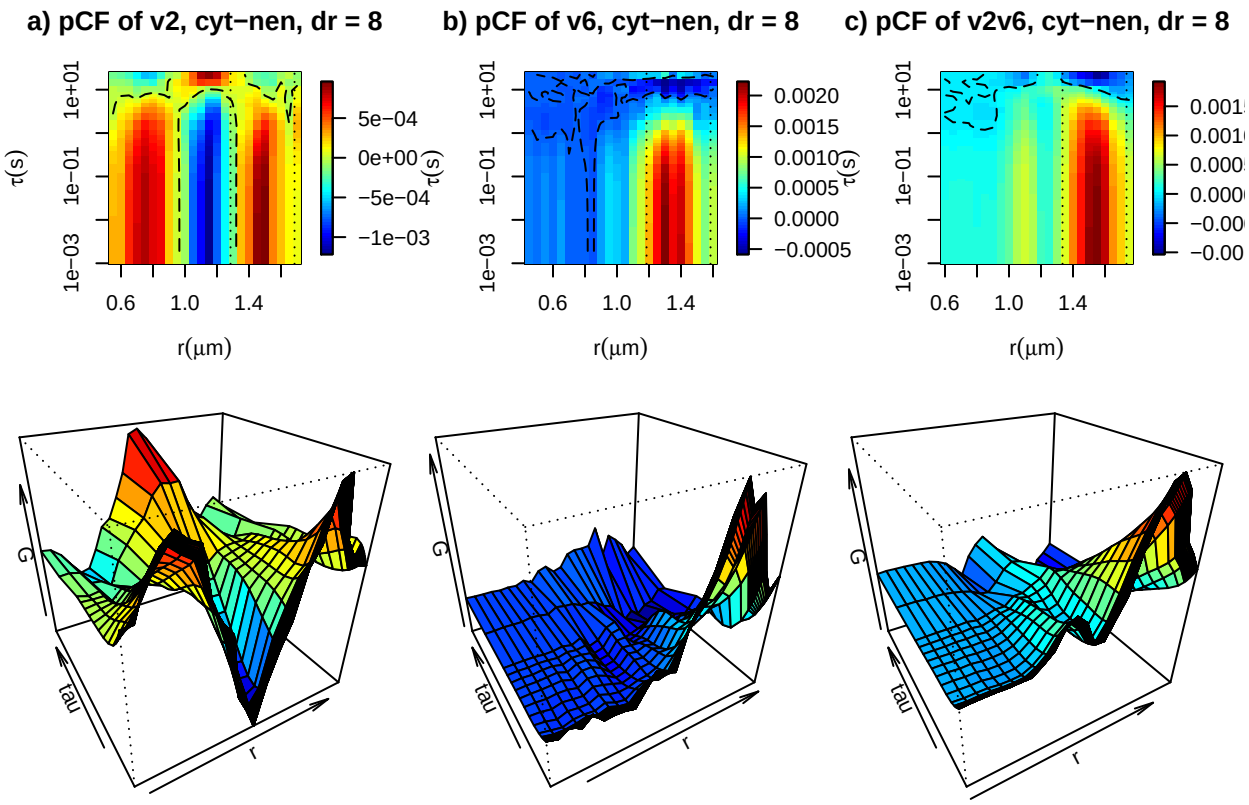

When the distance between the two points is smaller than the spatial resolution of the imaging system, in our example, in-between $r = 0.95\ \mu m$ and $r = 1.05\ \mu m$, positive correlation of the intensity fluctuations is observed at short correlation times $\tau < 0.01\ s$. As the distance increases, i.e. at $r = 1.2\ \mu m$, the time cross-correlation curve starts with very low (negative)-amplitude, increasig at a later time. This anticorrelation at short time is a signature of a single particle detected at a later times.

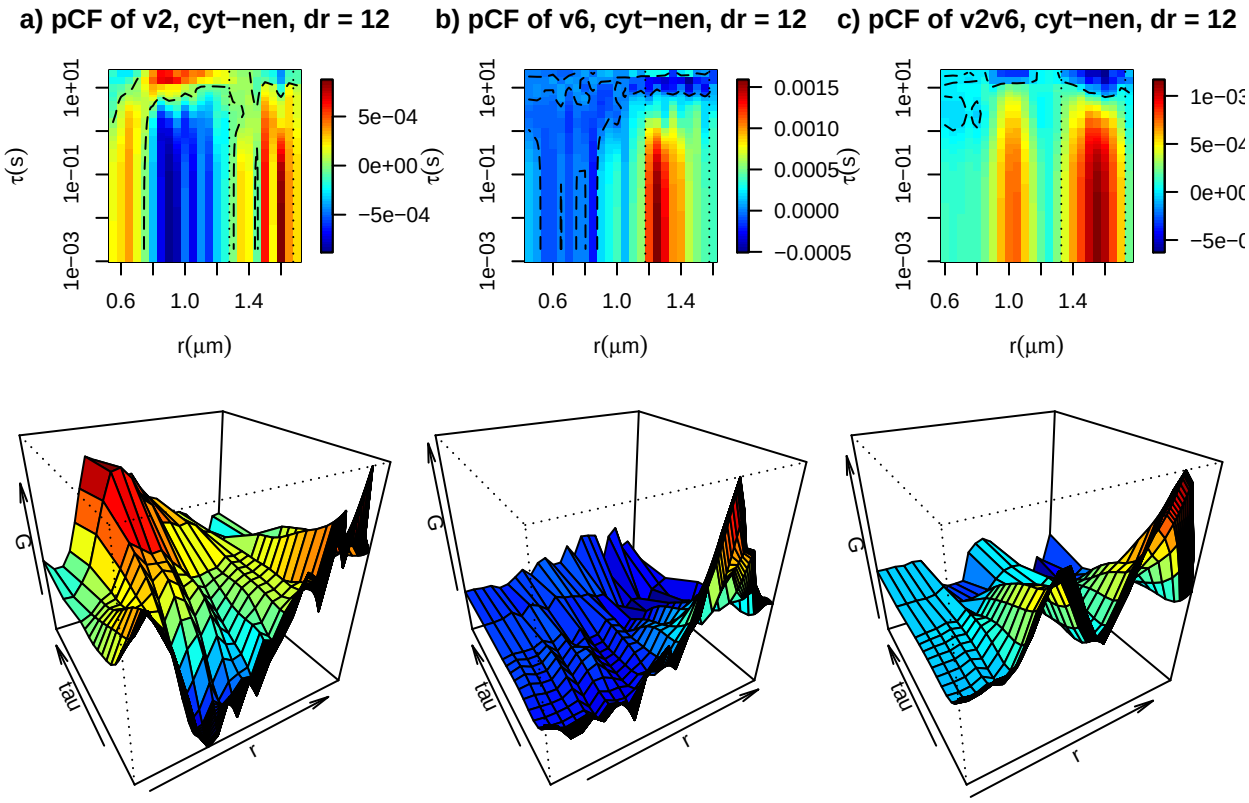
The pCF function can display a maximum of the correlation value at a given time (with $\tau \neq 0$) which depends on the diffusion coefficient. The shorter the time for the maximum of the pCF function, the faster the the mobility is.

Choosing $\delta r = 8$ pixels ($0.4\ \mu m$), or $\delta r = 12$ pixels ($0.6\ \mu m$) gives:

```
batchpCF <- function(dr, Range){
  nPoints <- dim(v2)[2]/2
  pCv2 <- pcf(img = v2, nPoints, type = "d", dr)
  pCv6 <- pcf(img = v6, nPoints, type = "d", dr)
  pCv2v6 <- pcf(img = v2v6, nPoints, type = "d", dr)
  ############################
  tau <- (1:nPoints)*(lineTime)
  r <- (1:dim(pCv2)[1])*pixelSize
```

```
  #Simplify data and interpolate
  pCv2.gs <- pCv6.gs<-pCv2v6.gs<-NULL
  for(px in 1:dim(pCv2)[1]){
    G <- simplifyFCS( pCv2[px,], tau)        #Simplify v2
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v2
    pCv2.gs <- rbind(pCv2.gs, g$y)
    G <- simplifyFCS( pCv6[px,], tau)        #Simplify v6
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v6
    pCv6.gs <- rbind(pCv6.gs, g$y)
    G <- simplifyFCS( pCv2v6[px,], tau)      #Simplify v2v6
    g <- smooth.spline(G$tau,G$g, df =3)     #Smooth v2v6
    pCv2v6.gs <- rbind(pCv2v6.gs, g$y)
  }
  tau.s <- g$x
  pCv2.dat <- list(y = tau.s, x = r[Range],  z = pCv2.gs[Range,])
  pCv6.dat <- list(y = tau.s, x = r[Range-2], z =pCv6.gs[Range-2,])
  pCv2v6.dat <- list(y = tau.s, x = r[Range+1], z = pCv2v6.gs[Range+1,])
  ######################
  par(mfrow = c(2,3))
  par(pin = c(1,1))
  image.plot(pCv2.dat,log = "y",
             xlab=expression(r(mu*m)), ylab= expression(tau(s)),
             main = paste("a) pCF of v2, cyt-nen, dr =",dr))
  contour(pCv2.dat, nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
  abline(v = c( 1.3, 1.7) - lineTime*dr, lty = 3)
  image.plot(pCv6.dat,log = "y",
             xlab=expression(r(mu*m)), ylab= expression(tau(s)),
             main = paste("b) pCF of v6, cyt-nen, dr =",dr))
  contour(pCv6.dat,nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
  abline(v = c(1.2, 1.6) - lineTime*dr, lty = 3)
  image.plot(pCv2v6.dat,log = "y",
             xlab=expression(r(mu*m)), ylab= expression(tau(s)),
             main = paste("c) pCF of v2v6, cyt-nen, dr =",dr))
  contour(pCv2v6.dat,nlevels = 1, add = T, lty = 5,levels = 0, drawlabels = F)
  abline(v = c(1.35, 1.75) - lineTime*dr, lty = 3)
  ####################
  par(pin = c(2.1,2.1))
  persp(pCv2.dat, col = compPerpsZval(pCv2.dat$z),
        xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
  persp(pCv6.dat, col = compPerpsZval(pCv6.dat$z),
        xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
  persp(pCv2v6.dat, col = compPerpsZval(pCv2v6.dat$z),
        xlab = "r", ylab = "tau", zlab = "G", phi = 30, theta = -30)
}
for(dr in c(8, 12)){
  batchpCF(dr, Range = 11:34)
}
```


a) pCF of v2, cyt−nen, dr = 8
b) pCF of v6, cyt−nen, dr = 8
c) pCF of v2v6, cyt−nen, dr = 8
τ(s)
1e+01
1e−01
1e−03
0.6
1.0
1.4
r(μm)
5e−04
0e+00
−5e−04
−1e−03
0.0020
0.0015
0.0010
0.0005
0.0000
−0.0005
G
tau
r




The "width" of the arc of positive cross-correlation depends on the dimension of the motion. The carpets of v2 computed at $\delta r = 8$ pixels (0.4 $\mu m$), or $\delta r = 12$ pixels (0.6 $\mu m$) show that, as $\delta r$ increases, the "width" of the arc of positive cross-correlation also increases. Hence, indicating single molecules reaching longer distances alongside the linescan.

### 1.3.4. Number & Brightness: molecular stoichiometry and relative concentration

As previously showed, the autocorrelation function can be used to measure molecular concentration and mobility within a certain region in the sample. The Number and Brightness (N&B) method, originally proposed by Qian and Elson in 1990 [7], is a time-independent technique that provides an estimate of molecular concentration and aggregation state (or stoichiometry), based on the statistical moments of the fluorescence intensity fluctuations. In other words, this tool allows to distinguish between two or more homo-oligomeric states of a molecule present in a given region in the sample (Brightness) while also providing a direct indicator of the molecules' relative abundance (Number) and spatial distribution.

For each pixel in an image or a scan line acquired in a confocal microscope, across a time series with $K$ total acquisitions, the signal intensity $k$ apparent mean value is given by

$$<k_i> = \frac{\Sigma_i k_i}{K}$$

Intensity fluctuations are mainly caused by particles constantly entering and leaving the observation volume over time. The apparent variance of these fluctuations in each pixel is given by

$$\sigma^2 = \frac{\Sigma_i (k_i - <k_i>)^2}{K}$$

The principle behind the N&B approach is that, when analyzing a series of fluorescence fluctuations across time, the more molecules present in the sample, the greater the mean intensity signal will be. On the other hand, the brighter these molecules are, the greater the variations around the mean value will be. Based on that, the number of molecules is proportional to the mean intensity, while the oligomerization state is

proportional to the variance of the intensity. Formally, the apparent Number ($N$) and apparent Brightness ($B$) are defined as

$N = \frac{<k>^2}{\sigma^2}$ $B = \frac{\sigma^2}{<k>}$

For any given pixel, $<k>$ is the mean value of fluorescence intensity and $\sigma^2$ is the variance.

While acquisition speed does not play a crucial role in this technique, if fast enough, it is possible to study aggregation dynamics. It also provides a means for excluding shot noise. Although N&B cannot solve multiple oligomeric species within the same pixel, the average particle number and brightness are calculated.

So far, we have assumed that the only factors contributing to $<k_i>$ and $\sigma^2$ is the molecular traffic created by molecules entering and leaving the observation volume, their concentration, aggregation state, quantum yield and the detector sensitivity. Nonetheless, there are several instrumentation-related parameters that must be considered prior to a quantitative analysis of the data. In theory, the signal should be zero when no photons are present, however, this isn't usually the case. The $Offset$, which is a constant quantity that depends on the detector configuration, corresponds to the average signal intensity across pixels when the detector is not exposed to any source of light. In order to obtain the real mean value of fluorescence intensity $<K>$, it is necessary to subtract the $Offset$ of the detector from the data.

$<K> = <k> - Offset$

Aditionally, the $S$ factor, a proportionality factor between the number of photons received and the number of electrons (signal) reported by the detector, must be calculated. The real mean value of fluorescence intensity $<K>$ is the result of the actual number of fluorophores inside the observation volume ($n$) and their actual molecular brightness ($\varepsilon$), both weighted by the proportionality factor S

$<K> = S\varepsilon n$

Matching the previous equations, we obtain

$<k> = S\varepsilon n + Offset$

On the other hand, the apparent variance $\sigma^2$ is the result of the sum of variance due to the fluorescence fluctuations ($\sigma_n^2$) and the variance due to detector noise ($\sigma_d^2$)

$(\sigma^2) = (\sigma_n^2) + (\sigma_d^2)$

While $\sigma_n^2$ depends exclusively on the proportionality factor $S$, and the real number and brightness of the molecules ($n$ and $\varepsilon$), $\sigma_d^2$ additionally depends on the detector readout noise ($\sigma_0^2$)

$\sigma_n^2 = S^2\varepsilon^2 n$

$\sigma_d^2 = S^2\varepsilon n + \sigma_0^2$

and we have

$\sigma^2 = S^2\varepsilon^2 n + S^2\varepsilon n + \sigma_0^2$

It is then necessary to know $\sigma_0^2$, which can be obtained by calculating the variance of the intensity values of the detector when exposed to no light sources. Altogether, considering these adjustments, the Number and Brightness equations can be rewritten as

$N = \frac{S\varepsilon n}{S\varepsilon + S}$ $\qquad$ $B = S\varepsilon + S$

or

$N = \frac{\varepsilon n}{\varepsilon + 1}$ $B = S(\varepsilon + 1)$

Note that, while $n$ depends on $\varepsilon$, $\varepsilon$ does not depend on $n$. By performing the proper substitutions, the real number and brightness equations can be expressed in terms of the $S$, $Offset$ and $\sigma_0^2$ factors:

$n = \frac{(<k> - Offset)^2}{\sigma^2 - \sigma_0^2 - S(<k> - offset)}$ $\qquad$ $\varepsilon = \frac{\sigma^2 - \sigma_0^2 - S(<k> - offset)}{S(<k> - Offset)}$

In order to calculate the $S$ factor, one can measure the detector variance ($\sigma_d^2$) in the absence of fluorescence fluctuations (i.e. without particles entering and leaving the observation volume). This can be done by acquiring a time series while the detector is exposed to a light source of constant intensity. Considering that $B = S(\varepsilon + 1)$, and in that the absence of fluorophores, the real brightness $\varepsilon$ is zero, the apparent brightness $B$ is reduced to the $S$ factor:

$B = S = \frac{\sigma^2 - \sigma_0^2}{<k> - Offset}$

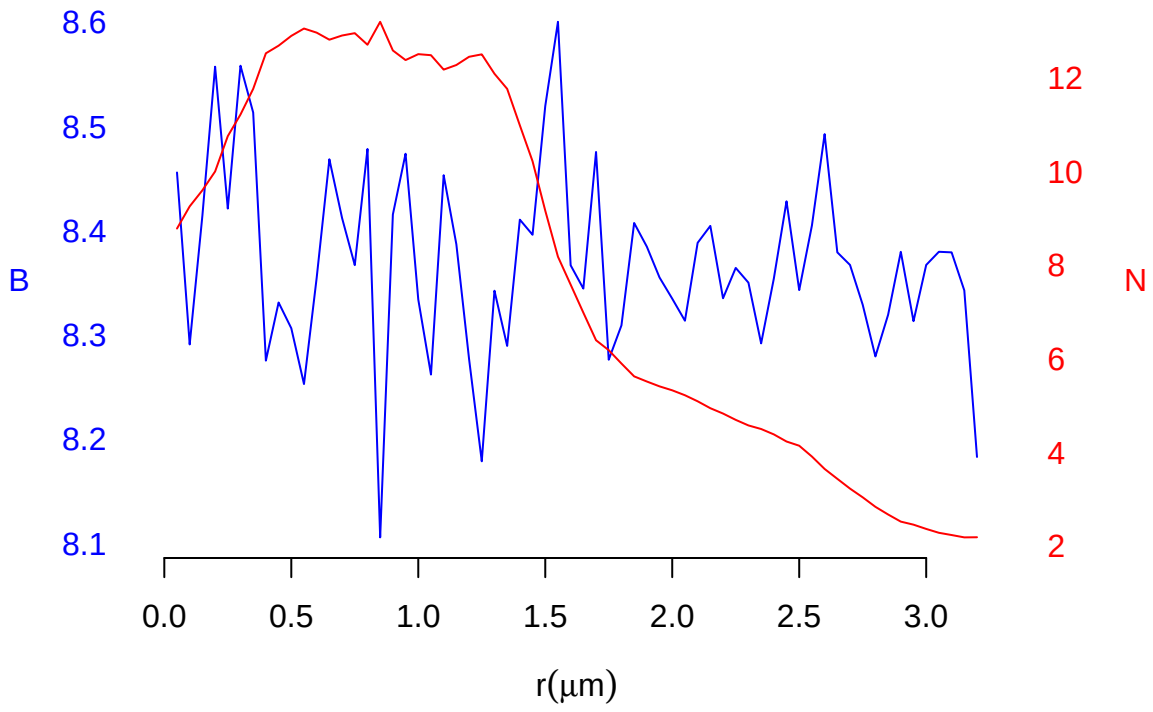
By isolating $\sigma^2$, we obtain an equation of the form $y = mx + b$, where the slope is the $S$ factor and the $y$ intercept is $\sigma_0^2$:

$\sigma^2 = S(<k> - Offset) + \sigma_0^2$

The nbline() function calculates the number and brightness values for each pixel across the scan line, which indicate the relative concentration of the fluorophore in each pixel and its oligomeric state, respectively. This function returns two vectors, each for the number and brightness values, respectively.

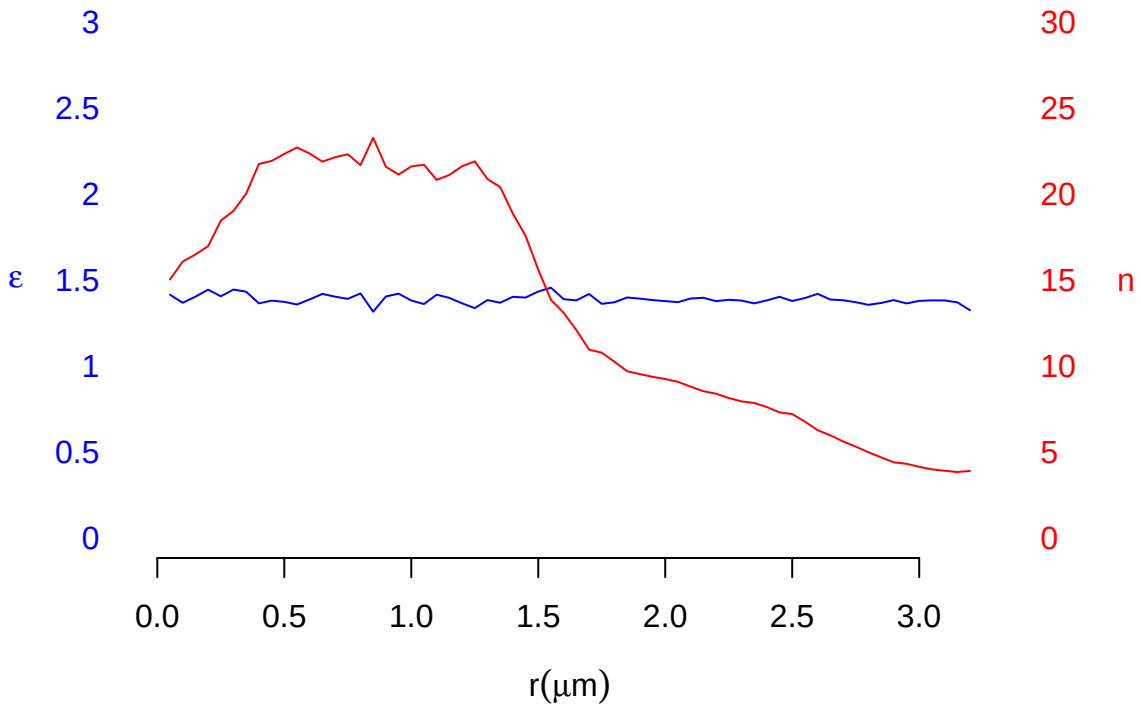
To compute the apparent Number and Brightness with the nbline() function, type:

```
nbv2 <- nbline(v2)
r <- (1:dim(v2)[1])*pixelSize
par(mar = c(5, 5, 3, 5))
plot(nbv2$Brightness~r, type = "l", col = "blue", axes = F, ann = F)
mtext(side = 2, text = axTicks(2), at = axTicks(2), col = "blue", line = 1, las = 1)
mtext(side = 2, text = "B", line = 3, col = "blue", las = 1)
axis(1)
mtext(side = 1, text = expression(r(mu*m)), line = 3, las = 1)
par(new = T)
plot(nbv2$ApparentNumber, type = "l", col = "red", axes = F, ann = F)
mtext(side = 4, text = axTicks(4), at = axTicks(4), col = "red", line = 1, las = 1)
mtext(side = 4, text = "N", line = 3, col = "red", las = 1)
```




To compute the real Number $n$ and Brightness $\varepsilon$ type:

```
par(mar = c(5, 5, 3, 5))
nbv2 <- nbline(img = v2, S=3.5, sigma0 = 1,offset = 0)
plot(nbv2$brightness~r, type = "l", ylim = c(0,3), col = "blue", axes = F, ann = F)
mtext(side = 2, text = axTicks(2), at = axTicks(2), col = "blue", line = 1, las = 1)
mtext(side = 2, text = expression(epsilon), line = 3, col = "blue", las = 1)
axis(1)
mtext(side = 1, text = expression(r(mu*m)), line = 3, las = 1)
par(new = T)
plot(nbv2$number, type = "l", col = "red", ylim = c(0,30), axes = F, ann = F)
mtext(side = 4, text = axTicks(4), at = axTicks(4), col = "red", line = 1, las = 1)
mtext(side = 4, text = "n", line = 3, col = "red", las = 1)
```




Notice that, in the case of a cell expressing Venus dimers, the brightness value remains almost constant (with minimal variations, i.e. ± 0.02 units), while the number data indicates approximately a one-third to a quarter reduction of protein abundance as the nuclear membrane is crossed through.

### 1.3.5 Pair Correlation of Molecular Brightness: creating an oligomer-specific molecular flux map

The Pair Correlation of Molecular Brightness approach combines the virtues of both the Pair Correlation Function and Number & Brightness methods. pCF allows to create a molecular flux map as it measures the time it takes for a particle to move from one position to another along a confocal line scan; though, it does not discriminate mobility based on oligomerization state. On the other hand, N&B can determine the concentration of fluorophores and their aggregation state based on the statistical moments of the fluorescence fluctuations; but it is mostly insensitive to particle dynamics. Originally proposed by Hinde and coworkers in 2016 [8], the pCOMB approach allows to distinguish between different oligomeric states of a given molecule coexisting in the same microenvironment while separately and specifically tracking each of the species mobility within the cellular compartments.

pCOMB calculates the temporal correlation between the brightness fluctuations. In this work, the N&B analysis is performed over a periodically scanned line ($x$,$t$) and, in order to gather every element needed for the pCOMB approach, a brightness carpet must be first generated, i.e. adding a temporal dimension for the brightness analysis to be then turn (through pCF) into a molecular flux map. The running average method can be used to generate the brightness carpet:

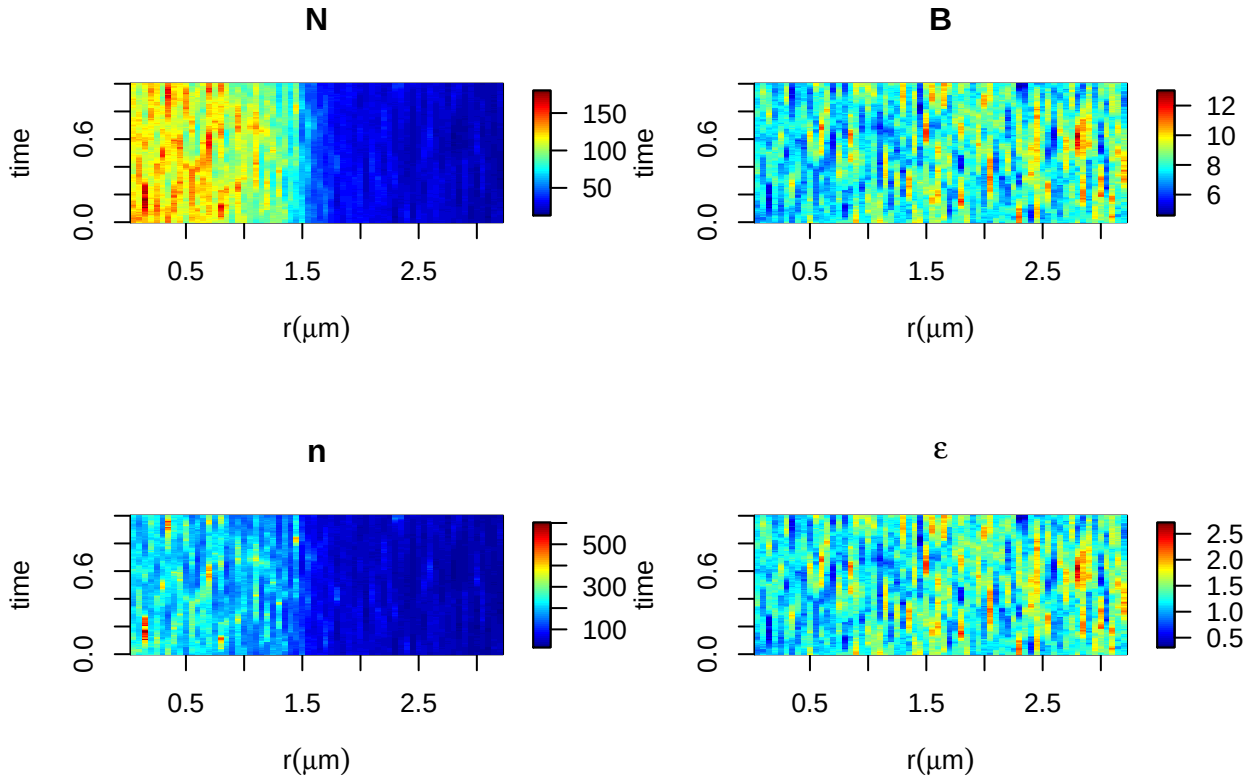
$B_r(t) = \frac{\sigma_r^2(t)}{<F_r(t)>}$

Where $B_r(t)$ is the moving apparent brightness at the $rth$ pixel across the line scan. For each scan time t, $B_r(t)$ considers a time interval $w = \delta t$, as the width of the moving average range. Within this temporal range, $\sigma_r^2(w) = \sigma_r^2(t_i, \ldots, t_i + w)$, and $F_r(w) = F_r(t_i, \ldots, t_i + w)$, with $t_i \in [0, \ldots, T - \delta t]$, with $T$ being the

total amount of line scans. The apparent brightness carpet will contain the same number of pixels across the line scan as the fluorescence intensity carpet, however, it will be w lines shorter.

To compute the Number and Brightness of V2V6 as function of time for one second, type:

```
appNB <- nbline(img = v2v6[,1:(1/lineTime)], wSigma = 100);
realNB <- nbline(img = v2v6[,1:(1/lineTime)], S=3.5, sigma0 = 1,offset = 0, wSigma = 100);
par(mfrow = c(2,2))
image.plot(x = r, z = appNB$ApparentNumber, xlab = expression(r(mu*m)),
                                            ylab = "time", main = "N")
image.plot(x = r, z = appNB$Brightness, xlab = expression(r(mu*m)),
                                        ylab = "time", main = "B")
image.plot(x = r, z = realNB$number, xlab = expression(r(mu*m)),
                                     ylab = "time", main = "n")
image.plot(x = r, z = realNB$brightness, xlab = expression(r(mu*m)),
                                         ylab = "time", main = expression(epsilon))
```




Now, in order to calculate pair correlation of the brightness fluctuations, a spatial component $(r)$ is introduced to the general correlation function

$G_B(\tau, \delta r) = \frac{<B(t,r)*B(t+\tau,r+\delta r)>}{<B(\tau,r)>*<B(t,r+\delta r)>} - 1$

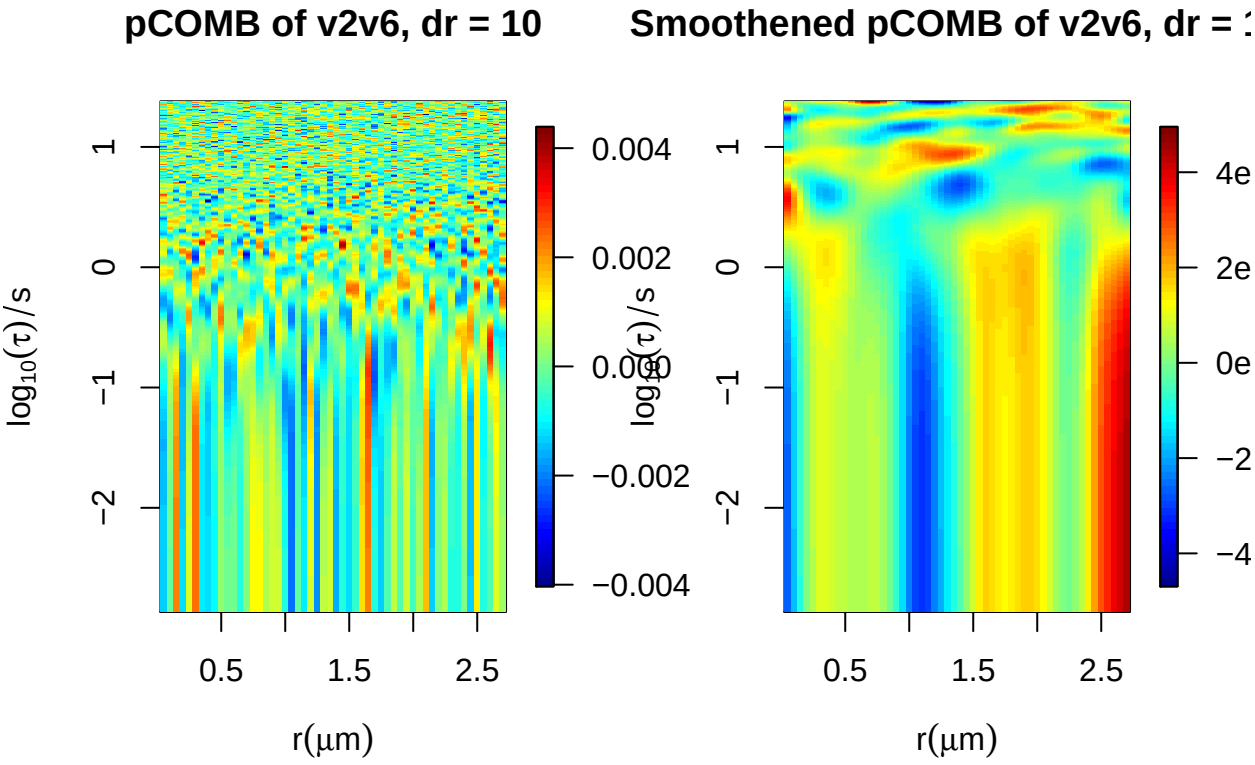
One should keep in mind that the real brightness must be first calculated before proceeding to create the brightness carpet, as it accounts for the photophysical characteristics of the detector, vital for proper data interpretation. To use the pcomb() function type

```
w <- 100
nPoints <- (dim(v2v6)[2]-w)/2
dr <- 10
```

```
pCOMB <- pcomb(v2v6, nPoints = nPoints, type = 'd', dr = dr, w = w)
Tau <- (1:nPoints)*(lineTime)
r<- (1:dim(pCOMB$pComb)[1])*pixelSize
par(mfrow = c(1,2))
image.plot(y =  log10(Tau), x = r, z = pCOMB$pComb,
           xlab=expression(r(mu*m)), ylab= expression(log[10](tau)/s),
           main = paste("pCOMB of v2v6, dr =",dr))

#Same pCOMB data, but smoothened

image.plot(y = log10(Tau), x=r,
           z = smoothCarpet( pCOMB$pComb,dfV = 10, dfH = 10),
           xlab=expression(r(mu*m)),
           ylab= expression(log[10](tau)/s),
           main = paste("Smoothened pCOMB of v2v6, dr =",dr))
```




Note te prescence of archs on the pCOMB carpet; they are indicative of single moving Venus oligomers.

# 2. Additional Methods

## 2.1 Preparation of biological samples

HEK-293 cells were cultured in cell culture treated 6-well polystyrene plates (Corning #3506) with DMEM/F-12 media (Gibco #11320-082) supplemented with 10% FBS (Corning # 35-010-CV), incubated at 37°C with 5% $CO_2$. Cells grown at $\sim$ 90% confluency were then transfected with two different plasmids containing dimeric and hexameric Venus (Addgene #29423 and #27813, respectively), using the Lipofectamine 2000™

transfection kit (Invitrogen #11668-019), and following the manufacturer guide. Transfected cells were incubated for 24h at 37ºC with 5% $CO_2$.

## 2.2 Microscopy data acquisition

Cy5 molecules were imaged in an Olympus FV1000 confocal microscope and a with a 60X objective, 1.3 NA and a pinhole of 140 $\mu$m for an excitation volume of 344 nm width. Excitation was provided by a 635 nm solid-state laser. Transfected cells were analyzed using an Olympus FV1000 confocal microscope, with a 60X water immersion objective, 1.2 NA. Fluorescence excitation was provided by a 488 nm laser at 0.1% power. Line-scans were performed across a 64-pixel line with a pixel size of 50 nm. Fluorescence intensity data was collected using the photon-counting mode, with a 12.5 $\mu s$ pixel dwell time, and line scan time of 1.925 ms.

## 3. References


[1] Tovar-Herrera, O. E., Rodríguez, M., Olarte-Lozano, M., Sampedro-Guerrero, J. A., Guerrero, A., Pinto-Cámara, R., ... & Segovia, L. (2018). Analysis of the Binding of Expansin Exl1, from Pectobacterium carotovorum, to Plant Xylem and Comparison to EXLX1 from Bacillus subtilis. ACS omega, 3(6), 7008-7018.

[2] Nagai, T., Ibata, K., Park, E. S., Kubota, M., Mikoshiba, K., & Miyawaki, A. (2002). A variant of yellow fluorescent protein with fast and efficient maturation for cell-biological applications. Nature biotechnology, 20(1), 87.

[3] Alber, F., Dokudovskaya, S., Veenhoff, L. M., Zhang, W., Kipper, J., Devos, D., ... & Sali, A. (2007). The molecular architecture of the nuclear pore complex. Nature, 450(7170), 695.

[4] Macara, I. G. (2001). Transport into and out of the nucleus. Microbiol. Mol. Biol. Rev., 65(4), 570-594.

[5] Hink, M. A., Griep, R. A., Borst, J. W., Van Hoek, A., Eppink, M. H., Schots, A., & Visser, A. J. (2000). Structural dynamics of green fluorescent protein alone and fused with a single chain Fv protein. Journal of Biological Chemistry, 275(23), 17556-17560.

[6] Digman, M. A., & Gratton, E. (2009). Imaging barriers to diffusion by pair correlation functions. Biophysical journal, 97(2), 665-673.

[7] Qian, H., & Elson, E. L. (1990). Distribution of molecular aggregation by analysis of fluctuation moments. Proceedings of the National Academy of Sciences, 87(14), 5479-5483.

[8] Hinde, E., Pandžić, E., Yang, Z., Ng, I. H., Jans, D. A., Bogoyevitch, M. A., ... & Gaus, K. (2016). Quantifying the dynamics of the oligomeric transcription factor STAT3 by pair correlation of molecular brightness. Nature communications, 7, 11047.