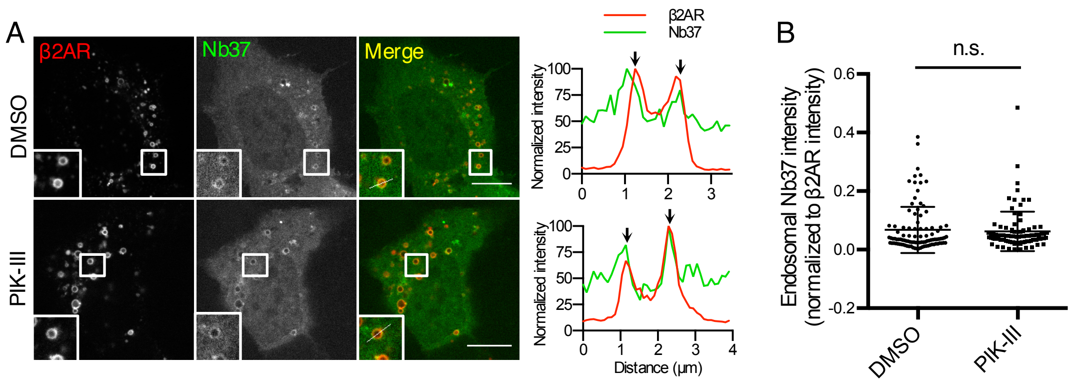
Supplementary Data

Endosomal phosphatidylinositol 3-kinase is essential for canonical GPCR signaling

Yasunori Uchida, Florentine U. Rutaganira, Damien Jullié, Kevan M. Shokat, and Mark von Zastrow

Supplementary Fig. 1.



Supplementary Fig. 1. VPS34 is not essential for G_s activation by β2AR at endosomes.
(A) Representative images of live HEK293 cells transiently expressing Flag-tagged β2AR and Nb37-GFP. Cells were pre-treated with PIK-III or DMSO for 1 h, and then treated with iso. Cells were imaged after 15 - 30 min of iso treatment by confocal microscopy. Insets show the boxed areas at higher magnification. Fluorescence intensity profiles of Flag-β2AR and Nb37-GFP from the lines in the insets were shown in the right graphs. Arrows indicate the enrichment of Nb37 signal in the limiting membrane of β2AR-containing endosomes. Scale bars, 10 *μ*m. (B) Quantification of the results in (A). The intensities of Nb37 and β2AR in endosome limiting membranes were calculated, and the Nb37 intensity was normalized to the β2AR intensity. *n* = 99 (DMSO) or 92 (PIK-III) endosomes from three independent experiments, n.s., not significant by two-tailed *t* test. Error bars mean SD.

Supplementary Text

Program script for measuring fluorescence intensity in endosomal limiting membranes

Bold letters show the part used for the actual analysis in this study. The 'width of linescan' was set to 3 pixels.

%This program allows a 3-clicks quantification of the "tubule index", enrichment of the marker in the tubule relative to the endosome.

%Rotative linescan to detect enrichment in the endosomal membrane relative to the tubule position

%

%Input is a Tif image +/- a region file

%Output is an excel file with the quantification and a .txt region file

%

%Two parameters can be set:

% size of the window that pops up "a", line 158

% number of steps for the rotative linescan "step" line 343

%

%This program won't run on mac Matlab because of Excel, minor modifications can be done to export files with the final matrixes. No other bug known so far.

%

%Run the program by entering "Tubulindex" on the command window, and select the .tif image.

%You can use the sliders to change the contrast, playing with them once the analysis is started will stop the program.

%If you want to create new regions, a message box will remind you to start by selecting a background region (user defined rectangle).

%A dialog box will then pop up for the user to enter the desired width of the linescan.

%I recommend entering an uneven number, but in all cases this must be an integer > 1 !

%

%The user then selects an endosome on the image, a window pops up to define more precisely the region of interest.

%The size of this window is fixed in the program, and can be changed by the user if necessary.

%If the structure is too big or too small, modify the parameter "a" accordingly (line 343).

%On the new window, the user will define the endosomal membrane + tubule in 3 clicks:

%

%-First click is the center of the endosome

%-Second click in the membrane of the endosome at the base of the tubule

%-Third click is the tip of the tubule

%The user then select another endosome, and so on.

%

%To stop selecting endosome:

%When you select the last endosome, use the RIGHT CLICK to select the center of the endosome. The program will recognize it as the last endosome.

%

%You can load the same endosomes using the Load ROIs button, just select the .txt file containing the info.

%This is particularly useful for comparison between channels.

%

%Quantification:

%

%Code will compute the average fluorescence in a donut centered around the first click.

%The inner diameter of the donut is the distance between first and second click – width of linescan/2.

%The outer diameter is distance between first and second click + width of linescan/2.

%This info is found on the excel sheet TubIndex, column "B", "Fluo Endo".

%

%Code does a rotative linescan. The number of steps for this linescan is defined is the program as 60 steps.

%You can modify the number of steps line 282, parameter "step".

%Basically, if you think of a clock, it will do a linescan of the donut for every second. 12 O'clock is at the opposite side of the tubule.

%The average fluorescence of the donut linescan is computed, and this information is reported on the sheet "Endo Linescan".

%Keep in mind that the first and last values are the same (noon/midnight).

%The program also plots the average values of this measurement +/- SEM.

%

%For the tubule, the program draws a line between the second and third click.

%Then it will draw a certain number of parallel lines centered on the user line.

%The number of line is the width of the linescan, they are 1 pixel distant.

%This is why I suggest uneven number, as the center of these lines is the one actually drawn by the user.

```
%The program does a linescan along every single line, compute the average fluorescence for every line, then the
average fluorescence of the different lines.
%This information is found on the excel sheet TubIndex, column “C”, “Fluo Tub”.
%Column D is the “Tubule index”, the ratio of column C/B.
%
%Once the computation is done
%green circle: circle drawn by the user, endosome
%red line: line drawn by the user, tubule
%white dots: outer diameter of the donuts, the linescan is made between the center of the circle and every single one
of these dots, starting from the opposite side of the red line.
%blue line: actual line used for the linescan, starting from the outer diameter of the donut.
%blue rectangle: region used for the linescan before transformation by the rotation matrix
%yellow rectangle: rectangle actually used for the linescan
%red dots: endpoints of the lines used for the linescans.
%Program will display a plot of the rotative linescan average fluorescence +/- SEM.
%
%
%If your image is named “MyImage”, the excel file will be named “MyImage_TubIndex.xlsx”.
%The text file containing the endosome coordinates “MyImage_endo.txt”.
%Do not run the program with preexisting files with the same names! It will overwrite.
%You’ll find average and SEM for both sheets.
%
%Last update on 12/13/2014
%
%Written by Damien Jullié

function Tubulindex(action)

%Open the image

if nargin == 0
```

```
[stk,stkd] = uigetfile('*.tif','Choose an image');

M = imread(stk);
M = double(M);

stk = stk(1:end-4);
frame = 1;
figure('name',['Play ',stk])
map = gray(256);


%Controls for image scaling and user interface

  high = double(max(max(max(M))));
  low = double(min(min(min(M))));
  uMax = find(M>high-100);
  uMin = find(M<low+100);
  MhPix = M;
  MhPix(uMax) = low;
  McPix = M;
  McPix(uMin) = high;
  hPix = double(max(max(max(MhPix)))); %the second highest pixel value in Movie
  cPix = double(min(min(min(McPix)))); %the second lowest pixel value in Movie
  uicontrol('style','slider','callback','Tubulindex scale',...
    'min',low,'max',high-3,'value',low,...
    'position',[240,15,120,15],'tag','scalelow')
  uicontrol('style','text','position',[210,15,30,15],'tag','low_text')
  uicontrol('style','text','position',[175,15,35,15],'string','Low')
  uicontrol('style','slider','callback','Tubulindex scale',...
    'min',low+3,'max',high,'value',high,...
    'position',[240,30,120,15],'tag','scalehigh')
  uicontrol('style','text','position',[210,30,30,15],'tag','high_text')
  uicontrol('style','text','position',[175,30,35,15],'string','High')
  uicontrol('style','checkbox','position',[370,15,60,15],...
      'string','HotPix','value',0,'tag','hotpix',...
      'callback','Tubulindex scale','userdata',[high,hPix])
  uicontrol('style','checkbox','position',[370,30,60,15],...
      'string','ColdPix','value',0,'tag','coldpix',...
      'callback','Tubulindex scale','userdata',[low,cPix])

 uicontrol('string','New ROIs','position',[30,50,80,15],...
      'tag','Endo','userdata',[],'callback','Tubulindex NewR')
  uicontrol('string','Load ROIs','position',[30,80,80,15],...
      'tag','Largeur','userdata',[],'callback','Tubulindex LoadR')

%%% drawing the figure

  set(gcf,'UserData',M,'keypressfcn','Tubulindex key','doublebuffer','on',...
    'colormap',map)
  u = image(M(:,:,frame),'cdatamapping','scaled','tag','movi');
  set(gca,'clim',[low,high],'tag','moviaxis')

  set(gca,'position',[0.13,0.15,0.9,0.75])
  h = title([stk,' Frame # = ',num2str(frame)],...
    'interpreter','none');
  set(h,'userdata',stk)
  axis image
  pixvalm
  scale

else
 eval(action)
end
```

```
%%% defining regions



function NewR
M = get(gcf,'userdata');
stk = get(get(gca,'title'),'userdata');

%size of the window for picking endosome

a = 20;



%defining background region

msgbox('Select the background region')
waitfor(gcf)

[Xback,Yback,Back,rect] = imcrop;
rect = round(rect);
bx = rect(1); by = rect(2); bw = rect(3); bh = rect(4);
X = [bx,bx,bx+bw,bx+bw,bx];
Y = [by,by+bh,by+bh,by,by];
line('XData',X,'YData',Y,'color','r')

%storing endosomes coordinates
endo = cat(2,[0,0,0],rect);

fluoback = sum(sum(M(by:by+bh,bx:bx+bw)))/((1+bw)*(1+bh));
```

```
%width of the linescan, >1
largeur = inputdlg({'width of the quantification line'},'Width');
largeur = str2num(largeur{1});

set(findobj('tag','Largeur'),'UserData',largeur);



button = 1;
endonum = 1;

while button == 1;

xy = ginput(1);
xy = round(xy);

limits = get(gca,'clim');

minx = xy(1)-a;
maxx = xy(1)+a +1;
miny = xy(2)-a;
maxy = xy(2)+a +1;



if xy(1)-a < 1
    minx = 1;
end
if xy(1)+a > size(M,2)
    maxx = size(M,2);
end
if xy(2)-a < 1
    miny = 1;
```

```
end
if xy(2)+a > size(M,1)
    maxy = size(M,1);
end

MiniM = M(miny:maxy,minx:maxx);

%closup on the selected endosome

figure('name',['Endosome ',num2str(endonum)])
set(gcf,'userdata',MiniM,'colormap',gray(256));
image(MiniM,'cdatamapping','scaled','tag','miniImage')
set(gca,'clim',limits);
line(a+1,a+1,'lineStyle','none','marker','+','markerEdgeColor','b')
drawnow

%three clics selection
if button == 1;

[x,y,button] = ginput(1);

x = round(x);
y = round(y);

line(x,y,'lineStyle','none','marker','+','markerEdgeColor','g')

[x2,y2] = ginput(1);
x2 = round(x2);
y2 = round(y2);

line(x2,y2,'lineStyle','none','marker','+','markerEdgeColor','g')
```

```
[x3,y3] = ginput(1);
x3 = round(x3);
y3 = round(y3);

line(x3,y3,'lineStyle','none','marker','+','markerEdgeColor','r')




diametre = sqrt((x-x2).^2 + (y-y2).^2);



line([x2,x3],[y2,y3],'LineStyle','-','Color','r','LineWidth',largeur)
rectangle('position',[x-diametre, y-diametre, 2*diametre, 2*diametre],...
    'curvature',[1 1],'LineStyle','-',...
    'LineWidth',largeur,'EdgeColor','g')

pause(0.5)
close

x = xy(1)-a + x -1;
y = xy(2)-a + y -1;

x2 = xy(1)-a + x2 -1;
y2 = xy(2)-a + y2 -1;

x3 = xy(1)-a + x3 -1;
y3 = xy(2)-a + y3 -1;

line([x2,x3],[y2,y3],'LineStyle','-','Color','r','LineWidth',2)
rectangle('position',[x-diametre, y-diametre, 2*diametre, 2*diametre],...
    'curvature',[1 1],'LineStyle','-',...
    'LineWidth',largeur,'EdgeColor','g')
```

```
endo = cat(1, endo, [endonum,x,x2,x3,y,y2,y3]);

endonum = endonum + 1;

end

end

%export the endosome coordinates file

dlmwrite([stk,'_endo.txt'], endo, 'delimiter','¥t');
set(findobj('tag','Endo'),'UserData',endo);

%run the quantification
QuantTub


%If load region is selected, pick a .txt file with endosome coordinates
%Loading regions

function LoadR


M = get(gcf,'userdata');

[f,p] = uigetfile('_syn.txt','Choose the region text file');
endo = dlmread([p,f],'¥t');

set(findobj('tag','Endo'),'UserData',endo);
```

```
largeur = inputdlg({'width of the quantification line'},'Width');
largeur = str2num(largeur{1});

set(findobj('tag','Largeur'),'UserData',largeur);



bx = endo(1,4); by = endo(1,5); bw = endo(1,6); bh = endo(1,7);
X = [bx,bx,bx+bw,bx+bw,bx];
Y = [by,by+bh,by+bh,by,by];
line('XData',X,'YData',Y,'color','r')

for i = 2:size(endo,1)


x = endo(i,2); x2 = endo(i,3); x3 = endo(i,4);
y = endo(i,5); y2 = endo(i,6); y3 = endo(i,7);

diametre = sqrt((x-x2).^2 + (y-y2).^2);

line([x2,x3],[y2,y3],'LineStyle','-','Color','r','LineWidth',2)
rectangle('position',[x-diametre, y-diametre, 2*diametre, 2*diametre],...
    'curvature',[1 1],'LineStyle','-',...
    'LineWidth',largeur,'EdgeColor','g')



end

QuantTub




%%% Quantification of the tubule index
```

```
function QuantTub



%Number of steps for the rotative linescan

step = 60;


M = get(gcf,'userdata');

stk = get(get(gca,'title'),'userdata');


children = get(gcf,'children');


endo = get(findobj(children,'tag','Endo'),'UserData');

largeur = get(findobj(children,'tag','Largeur'),'UserData');


bx = endo(1,4); by = endo(1,5); bw = endo(1,6); bh = endo(1,7);

X = [bx,bx,bx+bw,bx+bw,bx];

Y = [by,by+bh,by+bh,by,by];


fluoback = sum(sum(M(by:by+bh,bx:bx+bw)))/((1+bw)*(1+bh));


RotFluo2 = [];

Tubulindex = [];


for i = 2:size(endo,1)


RotFluo = endo(i,1);


x = endo(i,2); x2 = endo(i,3); x3 = endo(i,4);

y = endo(i,5); y2 = endo(i,6); y3 = endo(i,7);


diametre = sqrt((x-x2).^2 + (y-y2).^2);
```

```
[r,t] = meshgrid(1:size(M,2),1:size(M,1));

distance = sqrt((r-x).^2 + (t-y).^2);



%define the donut
diametreIn = diametre - largeur/2;
diametreOut = diametre + largeur/2;

maskIn = distance < diametreIn;
maskOut = distance < diametreOut;
mask = maskOut - maskIn;
Fluocircle = mask .* M;

% Rotative linescan


stepangle = 2*pi/step;
orangle = acos((x2-x)/diametre);

for k = 0 : step;


    xk = x + diametreOut * cos(pi - orangle + k*stepangle);
    yk = y + diametreOut * sin(pi - orangle + k*stepangle);



line(xk,yk,'lineStyle','none','marker','*','markerEdgeColor','w')

Rotline = improfile(Fluocircle,[x,xk],[y,yk]);
CleanRotline = find(Rotline ~= 0);
Rotline = mean(Rotline(CleanRotline),1) - fluoback;
RotFluo = cat(2,RotFluo,Rotline);
```

```
end

RotFluo2 = cat(1,RotFluo2,RotFluo);

%AvCircle is the backgroud substracted fluo in the donut (Uchida et al.
%2016)

AvCircle = sum(sum(Fluocircle))/sum(sum(maskOut-maskIn));
AvCircle = AvCircle - fluoback;



%quantification of the tubule florescence using a rotative rectangle
%This method is used to implement improfile accross multiple paralell lines

if (y2 ~= y3) & (x2 ~= x3)



pente1 = (y3 - y2)/(x3 - x2);

if x3 - x2 > 0

mmpc = sqrt(((largeur/2)^2)/(pente1^2 +1));

else

mmpc = - sqrt(((largeur/2)^2)/(pente1^2 +1));

end

x2 = x2 + mmpc;
y2 = y2 + pente1 * mmpc;
```

```
longueur = sqrt((x3-x2).^2 + (y3-y2).^2);

line([x2,x3],[y2,y3],'LineStyle','-','Color','c','LineWidth',4)

if y3 - y2 < 0

rotationAngle = acos((x3-x2)/longueur);
else
rotationAngle = - acos((x3-x2)/longueur);
end



rotationArray = [cos(rotationAngle), -sin(rotationAngle); sin(rotationAngle), cos(rotationAngle)];

centerx = (x3 - x2)/2;
centery = (y3 - y2)/2;

rectangle('position',[x2 + centerx - longueur/2, y2+ centery - largeur/2, longueur,
largeur],'LineStyle','-','LineWidth',2,'EdgeColor','b')

vertices2 = cat(1,[-longueur/2,-largeur/2],[longueur/2,-largeur/2],[-longueur/2,largeur/2],[longueur/2,largeur/2]);

rotrect2 = vertices2 * rotationArray;

rotrect2(:,1) = rotrect2(:,1) + x2 + centerx;
rotrect2(:,2) = rotrect2(:,2)+ y2 + centery;

%this is to draw the rotated rectangle

line([rotrect2(1,1),rotrect2(2,1)],[rotrect2(1,2),rotrect2(2,2)],'LineStyle','-','Color','y','LineWidth',2)
line([rotrect2(1,1),rotrect2(3,1)],[rotrect2(1,2),rotrect2(3,2)],'LineStyle','-','Color','y','LineWidth',2)
line([rotrect2(2,1),rotrect2(4,1)],[rotrect2(2,2),rotrect2(4,2)],'LineStyle','-','Color','y','LineWidth',2)
line([rotrect2(3,1),rotrect2(4,1)],[rotrect2(3,2),rotrect2(4,2)],'LineStyle','-','Color','y','LineWidth',2)




pente = (rotrect2(3,2) - rotrect2(1,2))/(rotrect2(3,1) - rotrect2(1,1));


b1 = rotrect2(1,2) - pente * rotrect2(1,1);
b2 = rotrect2(2,2) - pente * rotrect2(2,1);



Fluoline2 = [];


for j = 0 : (largeur-1)


xj1 = rotrect2(1,1) + ((rotrect2(3,1) - rotrect2(1,1))/(largeur-1))*j;
xj2 = rotrect2(2,1) + ((rotrect2(4,1) - rotrect2(2,1))/(largeur-1))*j;


yj1 = pente * xj1 + b1;
yj2 = pente * xj2 + b2;


%to draw the dots for the linescans


line(xj1,yj1,'lineStyle','none','marker','*','markerEdgeColor','r')
line(xj2,yj2,'lineStyle','none','marker','*','markerEdgeColor','r')


Fluoline = improfile(M,[xj1,xj2],[yj1,yj2]);
Fluoline = mean(Fluoline,1);
Fluoline2 = cat(1,Fluoline2,Fluoline);


end
```

```
else

%Vertical/horizontal lines (special cases)

    if y2 == y3
    if x3 > x2
    xj1 = x2 + largeur/2;
    else
    xj1 = x2 - largeur/2;
    end
    xj2 = x3;

    line([xj1,xj2],[y2,y3],'LineStyle','-','Color','c','LineWidth',4)

    Fluoline2 = [];

    for j = 0 : (largeur-1)

    yj1 = y2 -(largeur-1)/2 +j;
    yj2 = y3 -(largeur-1)/2 +j ;



    line(xj1,yj1,'lineStyle','none','marker','*','markerEdgeColor','r')
    line(xj2,yj2,'lineStyle','none','marker','*','markerEdgeColor','r')

    Fluoline = improfile(M,[xj1,xj2],[yj1,yj2]);
    Fluoline = mean(Fluoline,1);
    Fluoline2 = cat(1,Fluoline2,Fluoline);

    end
    else


    if y3 > y2
    yj1 = y2 + largeur/2;
    else
    yj1 = y2 - largeur/2;
    end
    yj2 = y3;

    line([x2,x3],[yj1,yj2],'LineStyle','-','Color','c','LineWidth',4)

    Fluoline2 = [];
    for j = 0 : (largeur-1)

    xj1 = x2 - (largeur-1)/2 +j;
    xj2 = x3 - (largeur-1)/2 +j;




    line(xj1,yj1,'lineStyle','none','marker','*','markerEdgeColor','r')
    line(xj2,yj2,'lineStyle','none','marker','*','markerEdgeColor','r')

    Fluoline = improfile(M,[xj1,xj2],[yj1,yj2]);
    Fluoline = mean(Fluoline,1);
    Fluoline2 = cat(1,Fluoline2,Fluoline);


    end


    end
end
```

```
FluoTub = mean(Fluoline2,1);

FluoTub = FluoTub - fluoback;


FluoTub = cat(2,endo(i,1),AvCircle,FluoTub,FluoTub/AvCircle);

Tubulindex = cat(1,Tubulindex,FluoTub);


end


%Calculate average and SEM accross the multiple endosomes


MoyFluo = mean(Tubulindex,1);

stdmoy=std(Tubulindex,0,1);

ET=stdmoy./sqrtm(size(Tubulindex,1));


MoyRot = mean(RotFluo2(:,2:end),1);

stdRot = std(RotFluo2(:,2:end),0,1);

ETRot = stdRot./sqrtm(size(RotFluo2,1));


normcirc = [0:step]/step;


figure

errorbar(normcirc,MoyRot, ETRot,'-og','MarkerEdgeColor','k');


MoyRot = cat(1,MoyRot,ETRot);

legendRot2 = {'Average';'StDev'};

MoyRot = cat(2,legendRot2,num2cell(MoyRot));



legendRot = cat(2,{'Endosome #'}, num2cell(normcirc));

RotFluo2 = cat(1,legendRot,cell(1,size(RotFluo2,2)),num2cell(RotFluo2),cell(1,size(RotFluo2,2)),MoyRot);
```

```
MoyFluo = cat(1,MoyFluo(2:end),ET(2:end));


legend = {'Endosome #','Fluo endo','Fluo Tub','Tubule Index'};
legend2 = {'Average';'StDev'};


legend2 = cat(2,legend2,num2cell(MoyFluo));


Final = cat(1,legend, cell(1,4),num2cell(Tubulindex),cell(1,4),legend2);



xlswrite([stk,'_TubIndex.xlsx'], Final,'TubIndex', 'A1')
xlswrite([stk,'_TubIndex.xlsx'], RotFluo2,'Endo Linescan', 'A1')







function scale
M = get(gcf,'userdata');
children = get(gcf,'children');
low = round(get(findobj(children,'tag','scalelow'),'value'));
high = round(get(findobj(children,'tag','scalehigh'),'value'));
%minlow = get(findobj(children,'tag','scalelow'),'min');
%maxhigh = get(findobj(children,'tag','scalehigh'),'max');
hotPix = get(findobj(children,'tag','hotpix'),'value');
hPix = get(findobj(children,'tag','hotpix'),'userdata');
coldPix = get(findobj(children,'tag','coldpix'),'value');
cPix = get(findobj(children,'tag','coldpix'),'userdata');
%M = get(gcf,'userdata');
if hotPix
    maxhigh = hPix(2);
```

```
else
     maxhigh = hPix(1);
end
if coldPix
     minlow = cPix(2);
else
     minlow = cPix(1);
end
if high > maxhigh
     high = maxhigh;
end
if low < minlow
     low = minlow;
end
if high == minlow+1
    high = high +1;
end
if low == maxhigh-1
    low = low-1;
end

set(gca,'clim',[low,high])
set(findobj(children,'tag','scalelow'),'max',high-1,'min',minlow,...
    'sliderstep',[1/(high-1-minlow),25/(high-1-minlow)],...
    'value',low)
set(findobj(children,'tag','low_text'),'string',num2str(low));
set(findobj(children,'tag','scalehigh'),'min',low+1,'max',maxhigh,...
    'sliderstep',[1/(maxhigh-(low+1)),25/(maxhigh - (low+1))],...
    'value',high)
set(findobj(children,'tag','high_text'),'string',num2str(high));
```

%End of the code
