capC-MAP Downstream Analysis using Python, pyGenomeTracks and bedtools

Chris Brackley

19 November 2018

This page shows an example of downstream analysis and plotting using various Python and command line based tools for Capture-C interaction profiles output from capC-MAP. It uses data obtained from GEO:GSE120666 as an example, with data from two experiments in two different cell types, each with the same four target viewpoints.

The tools used are:

- bedtools : a suit of tools for manipulation of bed and bedGraph files, available at https://github.com/arq5x/bedtools2, with documentation at https://bedtools.readthedocs.io
- pyGenomeTracks : a standalone Python program and library to plot beautiful genome browser tracks, available at https://github.com/deeptools/pyGenomeTracks

as well as the Python libraries MatPlotLib and Numpy.

Plot per experiment statistics

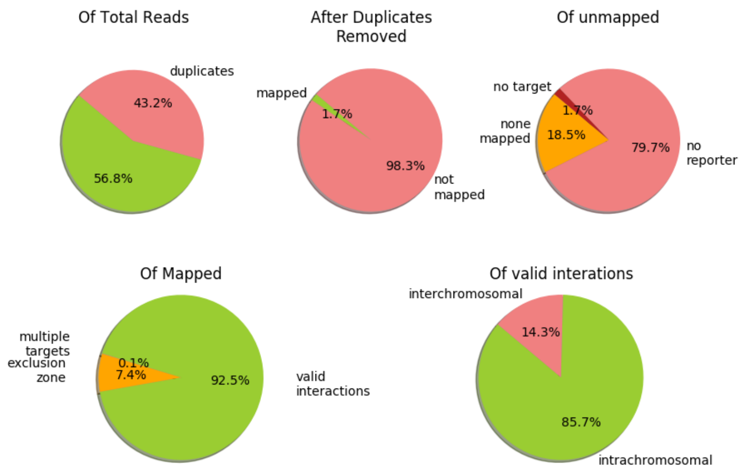
capC-MAP generates a file 'captured_report.dat' which contains information about read mapping, valid interactions, discarded reads etc. As part of the capC-MAP Python library there is a function to parse this file, which can be accessed from a Python script or interactive session when capC-MAP is installed on your system. The function returns a Python dictionary which can be used to generate plots to show the quality of the data.

For example the following can be run as a python script or in an interactive python session:

```
import numpy as np
import matplotlib.pyplot as plt
from capC import capCtools

stats = capCtools.read_report("data1/captured_report.dat")

f = plt.figure(figsize=(9, 6))

plt.subplot(2, 3, 1)
labels = '', 'duplicates'
sizes = [stats['reads'], stats['duplicates']]
colors = ['yellowgreen', 'lightcoral']
explode = (0, 0)
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Of Total Reads\n ')
plt.axis('equal')

plt.subplot(2, 3, 2)
labels = 'mapped', 'not\nmapped'
sizes = [stats['totalvalid']+stats['exclusion']+stats['multitarget'],
         stats['nonmapped']+stats['noreporter']+stats['notarget']]
```

```
colors = ['yellowgreen', 'lightcoral']
explode = (0,0)
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('After Duplicates\nRemoved')
plt.axis('equal')

plt.subplot(2, 3, 3)
labels = 'none\nmapped', 'no\nreporter', 'no target'
sizes = [stats['nonmapped'],stats['notarget'],stats['noreporter']]
colors = ['orange', 'lightcoral', 'firebrick']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Of unmapped\n ')
plt.axis('equal')

plt.subplot(2, 2, 3)
labels = 'valid\ninteractions','multiple\ntargets', 'exclusion\nzone'
sizes = [stats['totalvalid'],stats['multitarget'],stats['exclusion']]
colors = ['yellowgreen', 'lightcoral','orange']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=190,labeldistance=1.4)
plt.title('Of Mapped')
plt.axis('equal')

plt.subplot(2, 2, 4)
labels = 'intrachromosomal', 'interchromosomal'
sizes = [stats['validintra'],stats['validinter']]
colors = ['yellowgreen', 'lightcoral']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Of valid interations')
plt.axis('equal')

plt.subplots_adjust(hspace = 0.3,wspace = 0.5)
f.savefig("pyplots/plot_stats.png", bbox_inches='tight')
```

Generating the plot:



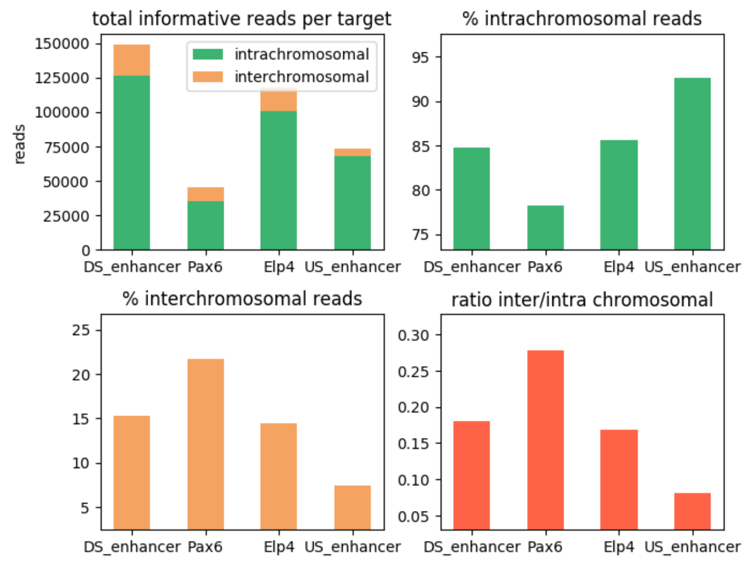
Plot per target statistics

capC-MAP also outputs statistics on each target in the file ‘captured_interactioncounts.dat’. Plots can be generated from this file, for example by running the following as a python script or in an interactive python session:

```
import numpy as np
import matplotlib.pyplot as plt

inter = {}
intra = {}
total = {}
within5Mb = {}
within1Mb = {}

with open("data1/captured_interactioncounts.dat", "r") as inf:
    for line in inf:
        a = line.split()
        if a[0] != "#":
            intra[a[0]] = int(a[1])
            inter[a[0]] = int(a[2])
            total[a[0]] = int(a[3])
            within5Mb[a[0]] = int(a[4])
            within1Mb[a[0]] = int(a[5])

N = len(inter)
ind = np.arange(N)
width = 0.5

f = plt.figure(figsize=(7.5, 6))

plt.subplot(2, 2, 1)
p1 = plt.bar(ind, np.fromiter(intra.itervalues(), dtype=float),width,
             label = 'intrachromosomal',color = 'mediumseagreen')
p2 = plt.bar(ind, np.fromiter(inter.itervalues(), dtype=float),width,
             label = 'interchromosomal',color = 'sandybrown',
             bottom = np.fromiter(intra.itervalues(), dtype=float))
plt.ylabel('reads')
plt.title('total informative reads per target')
plt.xticks(ind, intra.keys())
plt.legend()

plt.subplot(2, 2, 2)
percent_intra = 100*np.fromiter(intra.itervalues(), dtype=float) \
                  /np.fromiter(total.itervalues(), dtype=float)
p1 = plt.bar(ind, percent_intra ,width,color = 'mediumseagreen')
plt.ylim(max(min(percent_intra)-5,0), min(max(percent_intra)+5,100))
plt.title('% intrachromosomal reads')
plt.xticks(ind, intra.keys())

plt.subplot(2, 2, 3)
percent_inter = 100*np.fromiter(inter.itervalues(), dtype=float) \
                  /np.fromiter(total.itervalues(), dtype=float)
p1 = plt.bar(ind, percent_inter ,width,color = 'sandybrown')
```

```
plt.ylim(max(min(percent_inter)-5,0), min(max(percent_inter)+5,100))
plt.title('% interchromosomal reads')
plt.xticks(ind, intra.keys())

plt.subplot(2, 2, 4)
ratio = np.fromiter(inter.itervalues(), dtype=float) \
          /np.fromiter(intra.itervalues(), dtype=float)
p1 = plt.bar(ind, ratio ,width,color = 'tomato')
plt.ylim(max(min(ratio)-0.05,0), min(max(ratio)+0.05,1))
plt.title('ratio inter/intra chromosomal')
plt.xticks(ind, intra.keys())

plt.subplots_adjust(hspace = 0.3)
f.savefig("pyplots/plot_pertarget.png", bbox_inches='tight')
```

Generating the plot:

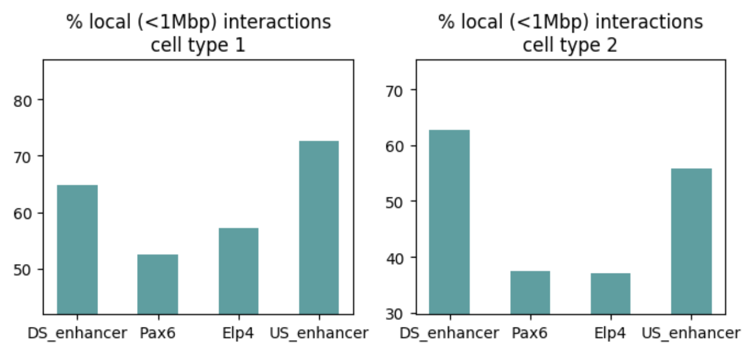


capC-MAP also reports the number of "local" interactions, defined either as those within 1Mbp or within 5Mbp of the target. A plot comparing data from two experiments can be generated in Python as follows:

```
import numpy as np
import matplotlib.pyplot as plt

total1 = {}
within1Mb1 = {}

f = plt.figure(figsize=(8, 3))

plt.subplot(1, 2, 1)

with open("data1/captured_interactioncounts.dat", "r") as inf:
    for line in inf:
        a = line.split()
        if a[0] != "#":
            total1[a[0]] = int(a[3])
            within1Mb1[a[0]] = int(a[5])
```

```
N = len(total1)
ind = np.arange(N)
width = 0.5

pc_local = 100*np.fromiter(within1Mb1.itervalues(), dtype=float) \
             /np.fromiter(total1.itervalues(), dtype=float)
p1 = plt.bar(ind, pc_local ,width,color = 'cadetblue')
plt.ylim(max(min(pc_local)*.8,0), min(max(pc_local)*1.2,100))
plt.title('% local (<1Mbp) interactions\ncell type 1')
plt.xticks(ind, total1.keys())

with open("data1/captured_interactioncounts.dat", "r") as inf:
    for line in inf:
        a = line.split()
        if a[0] != "#":
            total1[a[0]] = int(a[3])
            within1Mb1[a[0]] = int(a[5])
N = len(total1)
ind = np.arange(N)
width = 0.5

plt.subplot(1, 2, 2)

with open("data2/captured_interactioncounts.dat", "r") as inf:
    for line in inf:
        a = line.split()
        if a[0] != "#":
            total1[a[0]] = int(a[3])
            within1Mb1[a[0]] = int(a[5])
N = len(total1)
ind = np.arange(N)
width = 0.5

pc_local = 100*np.fromiter(within1Mb1.itervalues(), dtype=float) \
             /np.fromiter(total1.itervalues(), dtype=float)
p1 = plt.bar(ind, pc_local ,width,color = 'cadetblue')
plt.ylim(max(min(pc_local)*.8,0), min(max(pc_local)*1.2,100))
plt.title('% local (<1Mbp) interactions\ncell type 2')
plt.xticks(ind, total1.keys())

f.savefig("pyplots/plot_pertarget2.png", bbox_inches='tight')
```

This generates the plot:



Plot interaction profiles with different binning / smoothing parameters

The pyGenomeTracks package offers a simple command line tool for plotting bed and bedGraph files, reading from a configuration script. A gene track can be included by downloading a genes list from, e.g. UCSC, in BED12 format.

The following can be run at the command line to generate the configuration file 'tracks_compare_bins.ini':

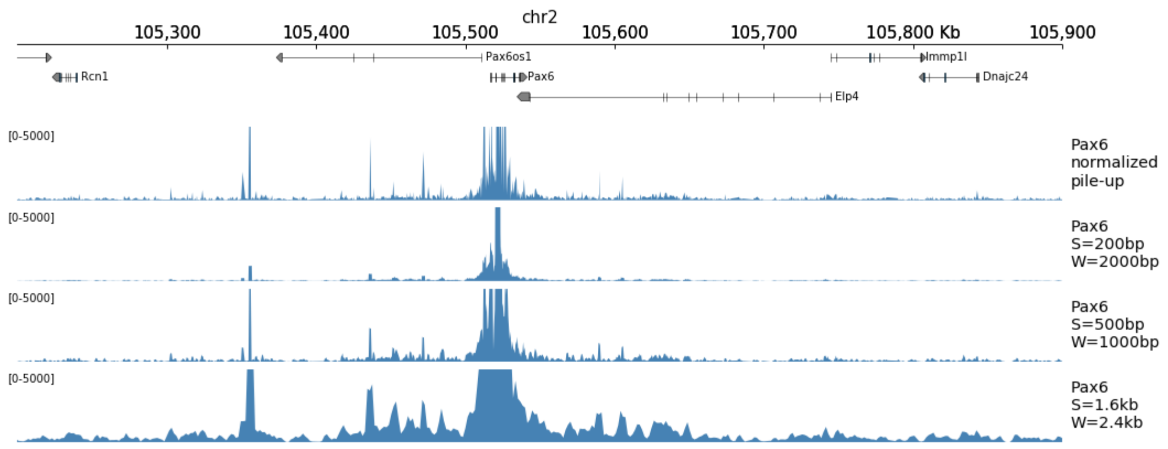
```
echo "
[x-axis]
#fontsize=10
where=top

[genes]
file = mm9_genes.bed.gz
title =
height = 2
fontsize = 10
file_type = bed
gene rows = 3

[spacer]
height = 0.5

[bedgraph file test]
file = data1/captured_normalizedpileup_Pax6.bdg
file_type = bedgraph
height = 2.5
title = Pax6 normalized pile-up
min_value = 0
max_value = 5e3
color = steelblue

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_200_2000_RPM_Pax6.bdg
file_type = bedgraph
height = 2.5
title = Pax6        S=200bp W=2000bp
min_value = 0
```

```
max_value = 5e3
color = steelblue

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_500_1000_RPM_Pax6.bdg
file_type = bedgraph
height = 2.5
title = Pax6          S=500bp W=1000bp
min_value = 0
max_value = 5e3
color = steelblue

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_1600_2400_RPM_Pax6.bdg
file_type = bedgraph
height = 2.5
title = Pax6         S=1.6kb W=2.4kb
min_value = 0
max_value = 5e3
color = steelblue

" > tracks_compare_bins.ini
```

Then the plot is generated using the command line:

```
pyGenomeTracks --tracks tracks_compare_bins.ini --region chr2:105,200,000-105,900,000 \
               --outFileName pyplots/plot_compare_bin.png
```

which gives the plot:



Plot interaction profiles from different targets

To include a track showing the positions of the targets, we need a bed file with 6 fields; this can be generated from the capC-MAP targets file using the following Unix command:

```
awk '{OFS="\t"; print $1,$2,$3,$4,".","+",0,0,0,0
     }' data1/targets.bed > pyplots/targets_forplots.narrowPeak
```

And then the configuration file 'tracks_all_targets.ini' can be generated with:

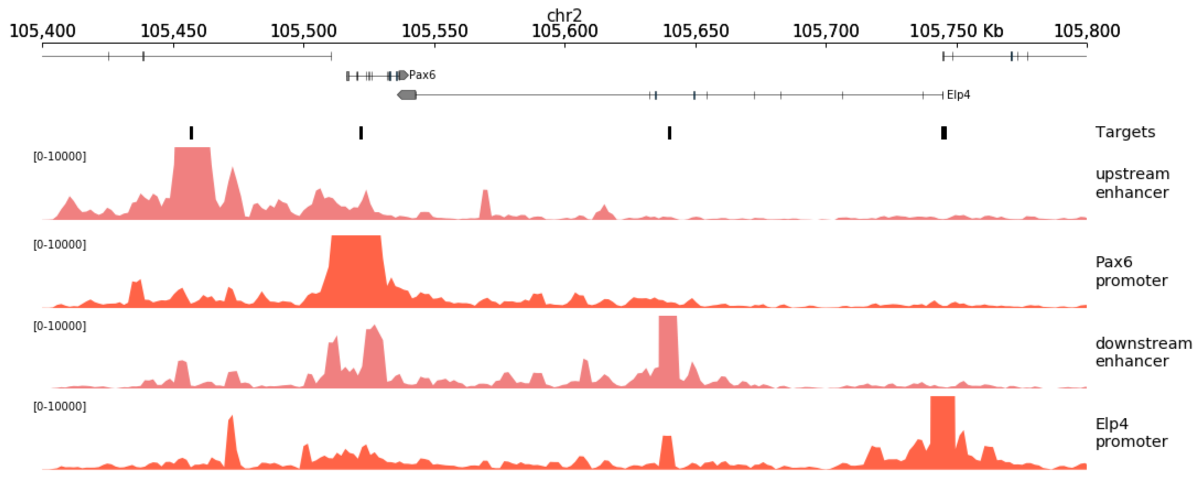
```
echo "
[x-axis]
#fontsize=10
where=top

[genes]
file = mm9_genes.bed.gz
title =
height = 2
fontsize = 10
file_type = bed
gene rows = 3

[spacer]
height = 0.5

[narrow]
file = pyplots/targets_forplots.narrowPeak
height = 0.5
track_type = narrow_peak
type = box
color = black
border color = black
title = Targets
show data range = no
show labels = no

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_1600_2400_RPM_US_enhancer.bdg
file_type = bedgraph
height = 2.5
title = upstream enhancer
min_value = 0
max_value = 10e3
color = lightcoral

[spacer]
height = 0.5

[bedgraph file test]
file = data1/captured_bin_1600_2400_RPM_Pax6.bdg
file_type = bedgraph
```

```
height = 2.5
title = Pax6 promoter
min_value = 0
max_value = 10e3
color = tomato

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_1600_2400_RPM_DS_enhancer.bdg
file_type = bedgraph
height = 2.5
title = downstream enhancer
min_value = 0
max_value = 10e3
color = lightcoral

[spacer]
height = 0.25

[bedgraph file test]
file = data1/captured_bin_1600_2400_RPM_Elp4.bdg
file_type = bedgraph
height = 2.5
title = Elp4 promoter
min_value = 0
max_value = 10e3
color = tomato

[spacer]
height = 0.25

" > tracks_all_targets.ini
```

Finally the plot is generated with the command:

```
pyGenomeTracks --tracks tracks_all_targets.ini --region chr2:105,400,000-105,800,000 \
               --outFileName pyplots/plot_all_targets.png
```

giving the plot:



Compare interaction profiles from two conditions / cell types

To compare, for example, two interaction profiles for the same target from different experiments, one can both (i) plot the profiles side-by-side, and (ii) generate a difference plot.

Using the bedtools software and the Unix tool awk, data from one bedGraph file can be subtracted from another using the following command lines:

```
bedtools unionbedg \
   -i data1/captured_bin_3000_6000_RPM_Pax6.bdg \
      data2/captured_bin_3000_6000_RPM_Pax6.bdg \
    | awk '{OFS="\t"; print $1,$2,$3,$4-$5}' > pyplots/diff_3000_6000_Pax6.bdg

bedtools unionbedg \
   -i data1/captured_bin_3000_6000_RPM_DS_enhancer.bdg \
      data2/captured_bin_3000_6000_RPM_DS_enhancer.bdg  \
    | awk '{OFS="\t"; print $1,$2,$3,$4-$5}' > pyplots/diff_3000_6000_DS_enhancer.bdg
```

Then the following command will generate the configuration file 'tracks_difference.ini':

```
echo "
[x-axis]
#fontsize=10
where=top

[genes]
file = mm9_genes.bed.gz
title =
height = 2
fontsize = 10
file_type = bed
gene rows = 3

[spacer]
height = 0.5

[bedgraph file test]
file = data1/captured_bin_3000_6000_RPM_Pax6.bdg
```

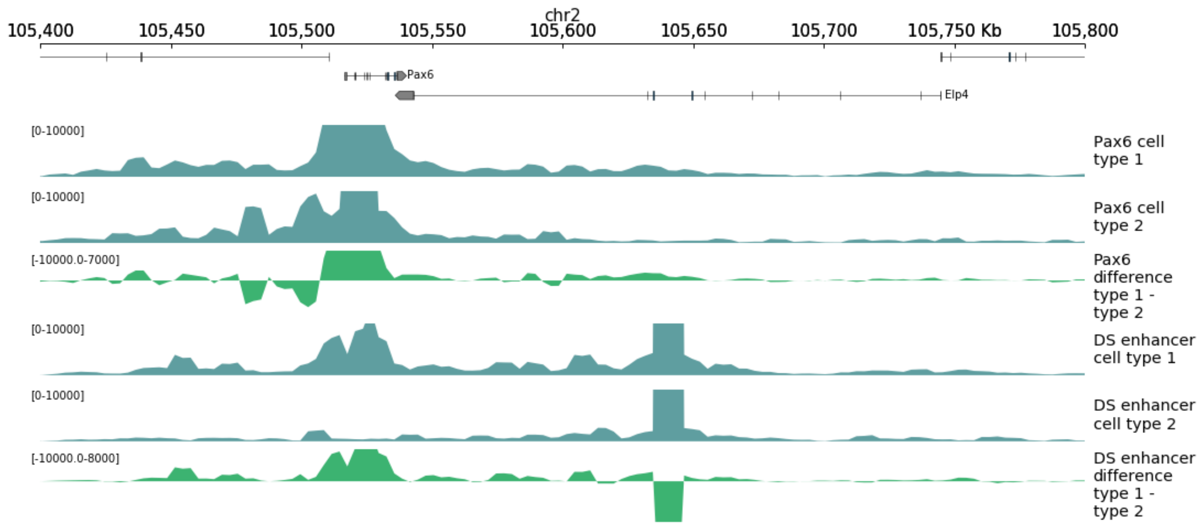
```
file_type = bedgraph
height = 1.75
title = Pax6 cell type 1
min_value = 0
max_value = 10e3
color = cadetblue

[spacer]
height = 0.5

[bedgraph file test]
file = data2/captured_bin_3000_6000_RPM_Pax6.bdg
file_type = bedgraph
height = 1.75
title = Pax6 cell type 2
min_value = 0
max_value = 10e3
color = cadetblue

[spacer]
height = 0.25

[bedgraph file test]
file = pyplots/diff_3000_6000_Pax6.bdg
file_type = bedgraph
height = 2.5
title = Pax6 difference   type 1 - type 2
min_value = -10e3
max_value = 7e3
color = mediumseagreen


[bedgraph file test]
file = data1/captured_bin_3000_6000_RPM_DS_enhancer.bdg
file_type = bedgraph
height = 1.75
title = DS enhancer cell type 1
min_value = 0
max_value = 10e3
color = cadetblue

[spacer]
height = 0.5

[bedgraph file test]
file = data2/captured_bin_3000_6000_RPM_DS_enhancer.bdg
file_type = bedgraph
height = 1.75
title = DS enhancer cell type 2
min_value = 0
max_value = 10e3
color = cadetblue
```

```
[spacer]
height = 0.25

[bedgraph file test]
file = pyplots/diff_3000_6000_DS_enhancer.bdg
file_type = bedgraph
height = 2.5
title = DS enhancer difference type 1 - type 2
min_value = -10e3
max_value = 8e3
color = mediumseagreen

" > tracks_difference.ini
```

Running the command:

```
pyGenomeTracks --tracks tracks_difference.ini --region chr2:105,400,000-105,800,000 \
               --outFileName pyplots/plot_diff.png
```

will then produce the plot:



Here the two difference plots show the profile from cell type 2 subtracted from cell type 1, meaning positive values mean there were more interactions in cell type 1, whereas negative values mean there were more interactions in cell type 2. The main difference shown is that the DS enhancer interact with Pax6 in cell type 1, but mainly interacts locally in cell type 2.