fga3_GROUP_re377

April 25, 2022

1 Genomics Assignment 3

Ralph Estanboulieh

1.0.1 Importing relevant packages

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


plt.rcParams['figure.dpi'] = 300             # for high quality figures
plt.rcParams["font.family"] = "Helvetica"
from scipy.stats import gaussian_kde         # for the KDE plots
from matplotlib import cm                    # for fancy colormaps
import re
```

1.1 Figure 1

[1064]:
```
# reading the tables in Pandas dataframes
table1 = pd.read_csv("table1.csv")
table2 = pd.read_csv("table2.csv")
table3 = pd.read_csv("table3.csv")

# turn the percentages into floats
table1["male_percentage"] = [float(x[:-1]) for x in table1["male_percentage"].
 ↪tolist()]

# get the curated subtypes
subtypes = ["pan_cancer"] + table1["curated_subtype"].tolist()
N_subtypes = len(subtypes)
```

Fixing Table 2 The ICD descriptions do not always match the metastasis count columns. We will have to fix that.

NOTE: It seems like the fix makes it even more different than the original figure. Do NOT run this code unless you have to.

[978]:
```
met_sites = table3.iloc[:, 0].tolist()
icd_codes = [re.findall(r"\'([^\\']*)\'", x) for x in table3.iloc[:, 1]]
icd_codes_to_met_sites = {code:site for codes, site in zip(icd_codes,␣
 ↪met_sites) for code in codes}

# had to do the matching manually because the names are not always consistent

met_site_to_met_count = {'ADRENAL_GLAND': 'met_count:Adrenal.Gland',
 'BILIARY_TRACT': 'met_count:Biliary.tract',
 'BLADDER_OR_URINARY_TRACT': 'met_count:Bladder/UT',
 'BONE': 'met_count:Bone',
 'BOWEL': 'met_count:Bowel',
 'BREAST': 'met_count:Breast',
 'CNS_BRAIN': 'met_count:CNS/Brain',
 'GENITAL_FEMALE': 'met_count:Female.Genital',
 'GENITAL_MALE': 'met_count:Male.Genital',
 'HEAD_AND_NECK': 'met_count:Head.and.Neck',
 'KIDNEY': 'met_count:Kidney',
 'LIVER': 'met_count:Liver',
 'LUNG': 'met_count:Lung',
 'LYMPH': 'met_count:Distant.LN',
 'MEDIASTINUM': 'met_count:Mediastinum',
 'OTHER': 'met_count:Unspecified',
 'OVARY': 'met_count:Ovary',
 'PERIPHERAL_NERVOUS_SYSTEM': 'met_count:PNS',
 'PERITONEUM': 'met_count:Intra-Abdominal',
 'PLEURA': 'met_count:Pleura',
 'SKIN': 'met_count:Skin'}

icd_codes_to_met_count = {code:met_site_to_met_count[met_site] for code,␣
 ↪met_site in icd_codes_to_met_sites.items()}

# set the met_count:X values to 0

table2.loc[:, table2.columns.str.contains("met_count:")] = 0

for index, row in table2.iterrows():
    row_icd_desc = row["icd_description"]
    if not pd.isna(row_icd_desc):
        row_codes = re.findall(r".{3}\.\d\d?", row_icd_desc)
        row_met_counts = [icd_codes_to_met_count.get(c, "met_count:
 ↪Unspecified") for c in row_codes]
        for met_count in row_met_counts:
            table2.loc[index, met_count] = table2.loc[index, met_count] + 1
```

1.1.1 Age, OS, sex, sample type, and metastasis burden

[1065]:
```
def text_plot(subtype, axis):
    tumor_type = "Pan-cancer" if subtype == "pan_cancer" else␣
 ↪table1[table1["curated_subtype"]==subtype]["curated_subtype_display"].item()
    abbrev = "PanCan" if subtype == "pan_cancer" else␣
 ↪table1[table1["curated_subtype"]==subtype]["curated_subtype_abbr"].item()
    n = str(len(table2)) if subtype == "pan_cancer" else␣
 ↪str(len(table2[table2["curated_subtype"]==subtype]))

    color = "black" if subtype == "pan_cancer" else␣
 ↪table1[table1["curated_subtype"]==subtype]["color_subtype"].item()

    axis.text(1, 0, tumor_type, horizontalalignment = 'right', color = color)
    axis.text(2.25, 0, abbrev, horizontalalignment = 'right', color = color)
    axis.text(3, 0, n, horizontalalignment = 'right', color = color)

    axis.set_xlim([0,3])
    axis.axis('off')


def age_plot(subtype, axis):
    table = table2 if subtype == "pan_cancer" else␣
 ↪table2[table2["curated_subtype"] == subtype]

    data = table["seq_report_age"].to_numpy()
    data = data[~np.isnan(data)]/365
    dens = gaussian_kde(data)
    median = np.median(data)

    x = np.arange(18, 90, 1)
    axis.plot([median, median], [0, dens(median)[0]], linewidth = 0.5, color =␣
 ↪"red")
    axis.plot(x, dens(x), color = "k", linewidth = 0.5)
    axis.axis('off')

def kaplan_meier_survival(data):
    x_range = [0] + sorted(list(set(data[data < 5]))) + [5]
    Di = np.histogram(data, x_range)[0]
    Ni = np.array([(data >= x).sum() for x in x_range])
    S = [1 - Di[0]/Ni[0]]
    [S.append(S[-1] * (1 - Di[idx]/Ni[idx])) for idx, x in enumerate(x_range[1:
 ↪])]
    S = np.array(S)
    median = x_range[np.argmax(S<=0.5)]
    return x_range, S, median
```

```
def os_plot(subtype, axis):
    table = table2 if subtype == "pan_cancer" else␣
 ↪table2[table2["curated_subtype"] == subtype]

    data = table["os_days"].to_numpy()
    data = (data + 1)/365
    data[np.isnan(data)] = 100
    xsurv, ysurv, median = kaplan_meier_survival(data)
    median = np.median(data)

    axis.plot([median, median], [0, 0.5], linewidth = 1, color = "red")
    axis.plot(xsurv, ysurv, color = "k", linewidth = 0.5)
    axis.set_ylim([0, 1])
    axis.axis('off')

def sex_plot(subtype, axis):
    table = table2 if subtype == "pan_cancer" else␣
 ↪table2[table2["curated_subtype"] == subtype]

    male_perc = (table["sex"] == "Male").sum()/len(table) * 100
    fem_perc = 100 - male_perc

    axis.barh([0], [fem_perc], color = "khaki")
    axis.barh([0], [male_perc], left = [fem_perc], color = "slategray")
    axis.set_xlim([0,100])
    axis.axis('off')

def sample_type(subtype, axis):
    table = table2 if subtype == "pan_cancer" else␣
 ↪table2[table2["curated_subtype"] == subtype]

    met = len(table[table["sample_type"] == "Metastasis"])
    primary = len(table[table["met_count"] == 0])
    primary_met = len(table) - primary - met

    axis.barh(0, primary, left = 0, color = "skyblue")
    axis.barh(0, primary_met, left = primary, color = "tab:blue")
    axis.barh(0, met, left = primary + primary_met, color = "tab:red")
    axis.axis('off')

def metastasis_burden(subtype, axis):
    table = table2 if subtype == "pan_cancer" else␣
 ↪table2[table2["curated_subtype"] == subtype]

    met_burden = np.array(table["met_count"])
    met_burden = np.histogram(met_burden, [0,1,2,3,4,5,6,np.inf])[0]
    met_burden = met_burden/sum(met_burden)
```

```
    starts = np.insert(np.cumsum(met_burden), 0, 0)

    cmap = cm.get_cmap('Reds', 8)
    for idx, (burden, start) in enumerate(zip(met_burden, starts)):
        axis.barh(0, burden, left = start, color = cmap(idx+1))
    axis.set_xlim([0, 1])
    axis.axis('off')
```

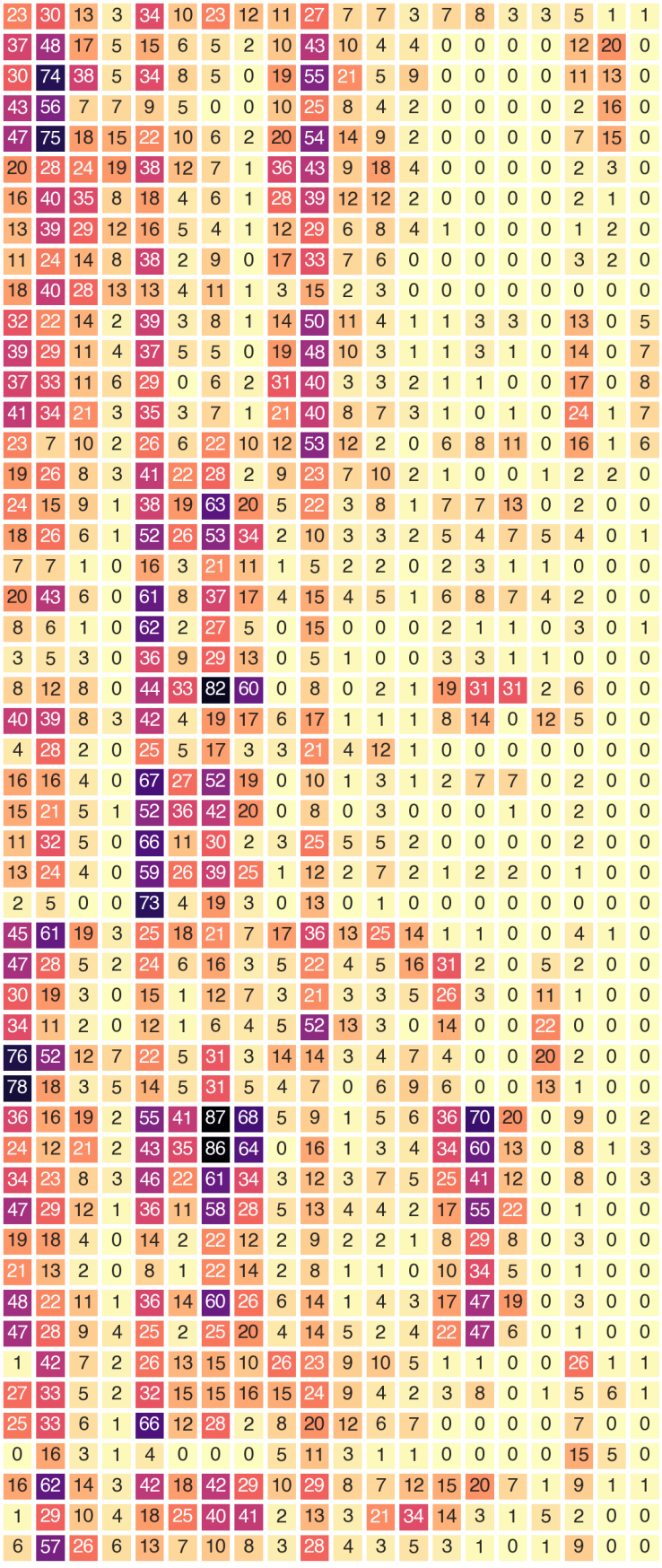
1.2 Metastasis site heatmap and stacked bar plots

[1066]:
```
met_count_cols = table2.columns[table2.columns.str.contains("met_count:")].
↪to_list()

met_count_table = table2.filter(like = "met_count:").copy().ge(1).astype(int)
met_count_table["n_samples"] = 1
met_count_table["curated_subtype"] = table2["curated_subtype"]
met_count_table = met_count_table.groupby("curated_subtype").aggregate(func =␣
↪np.sum)

pan_cancer_met_count = met_count_table.sum(0)
pan_cancer_met_count.name = "pan_cancer"
met_count_table = met_count_table.append(pan_cancer_met_count)

met_count_table = met_count_table.loc[subtypes,:]

def met_site_heatmap_plot(axis):
    data = met_count_table.to_numpy()
    data = (data / data[:, -1:] * 100).astype('int32')
    sns.heatmap(data[:, 1:-1], cbar = False,
                annot = True, cmap = "magma_r",
                linewidths = 1.5, annot_kws = {"fontsize":8}, ax = axis)
    axis.axis('off');

def unspec_met_heatmap_plot(axis):
    data = met_count_table.to_numpy()
    data = (data / data[:, -1:] * 100).astype('int32')
    sns.heatmap(data[:, 0:1], cbar = False,
                annot = True, cmap = sns.color_palette("ch:s=-.2,r=.6",␣
↪as_cmap=True),
                linewidths = 1.5, annot_kws = {"fontsize":8}, ax = axis)
    axis.axis('off');

met_site_colors = ["#999966", "#c82ffe",
 "#6228cb", "#5b384f", "#00fe99",
 "#066633", "#996534", "#679701",
```

```
 "#ca0001", "#35cbfe", "#39699a",
 "#663367", "#fd6566", "#993332",
 "#993332", "#fdcc03", "#660001",
 "#96cc34", "#0e9247", "#e6308e"]

def met_site_bar_plot(subtype, axis):
    counts = met_count_table.loc[subtype, met_count_cols[1:]].to_numpy()
    counts = counts/counts.sum()
    starts = np.insert(np.cumsum(counts), 0, 0)
    for count, start, color in zip(counts, starts, met_site_colors):
        axis.barh(0, width = count, left = start, color = color)
    axis.axis('off')
    axis.set_xlim([0, 1])

subtype_colors = table1['color_subtype'].to_list()

def met_cancer_bar_plot(met_subtype, axis):
    counts = met_count_table.loc[:, met_subtype].to_numpy()[1:]
    counts = counts/counts.sum()
    starts = np.insert(np.cumsum(counts), 0, 0)
    for count, start, color in zip(counts, starts, subtype_colors):
        axis.bar(0, height = count, bottom = start, color = color)
    axis.axis('off')
    axis.set_ylim([0, 1])
    axis.invert_yaxis()
```

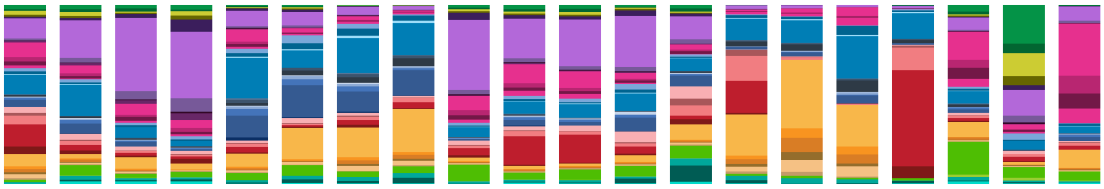
1.2.1 Testing the plotting functions

[1067]:
```
fig, ax = plt.subplots(1, 1, figsize = (12*21/51, 12))
met_site_heatmap_plot(ax)
```


23 30 13 3 34 10 23 12 11 27 7 7 3 7 8 3 3 5 1 1
37 48 17 5 15 6 5 2 10 43 10 4 4 0 0 0 0 12 20 0
30 74 38 5 34 8 5 0 19 55 21 5 9 0 0 0 0 11 13 0
43 56 7 7 9 5 0 0 10 25 8 4 2 0 0 0 0 2 16 0
47 75 18 15 22 10 6 2 20 54 14 9 2 0 0 0 0 7 15 0
20 28 24 19 38 12 7 1 36 43 9 18 4 0 0 0 0 2 3 0
16 40 35 8 18 4 6 1 28 39 12 12 2 0 0 0 0 2 1 0
13 39 29 12 16 5 4 1 12 29 6 8 4 1 0 0 0 1 2 0
11 24 14 8 38 2 9 0 17 33 7 6 0 0 0 0 0 3 2 0
18 40 28 13 13 4 11 1 3 15 2 3 0 0 0 0 0 0 0 0
32 22 14 2 39 3 8 1 14 50 11 4 1 1 3 3 0 13 0 5
39 29 11 4 37 5 5 0 19 48 10 3 1 1 3 1 0 14 0 7
37 33 11 6 29 0 6 2 31 40 3 3 2 1 1 0 0 17 0 8
41 34 21 3 35 3 7 1 21 40 8 7 3 1 0 1 0 24 1 7
23 7 10 2 26 6 22 10 12 53 12 2 0 6 8 11 0 16 1 6
19 26 8 3 41 22 28 2 9 23 7 10 2 1 0 0 1 2 2 0
24 15 9 1 38 19 63 20 5 22 3 8 1 7 7 13 0 2 0 0
18 26 6 1 52 26 53 34 2 10 3 3 2 5 4 7 5 4 0 1
7 7 1 0 16 3 21 11 1 5 2 2 0 2 3 1 1 0 0 0
20 43 6 0 61 8 37 17 4 15 4 5 1 6 8 7 4 2 0 0
8 6 1 0 62 2 27 5 0 15 0 0 0 2 1 1 0 3 0 1
3 5 3 0 36 9 29 13 0 5 1 0 0 3 3 1 1 0 0 0
8 12 8 0 44 33 82 60 0 8 0 2 1 19 31 31 2 6 0 0
40 39 8 3 42 4 19 17 6 17 1 1 1 8 14 0 12 5 0 0
4 28 2 0 25 5 17 3 3 21 4 12 1 0 0 0 0 0 0 0
16 16 4 0 67 27 52 19 0 10 1 3 1 2 7 7 0 2 0 0
15 21 5 1 52 36 42 20 0 8 0 3 0 0 0 1 0 2 0 0
11 32 5 0 66 11 30 2 3 25 5 5 2 0 0 0 0 2 0 0
13 24 4 0 59 26 39 25 1 12 2 7 2 1 2 2 0 1 0 0
2 5 0 0 73 4 19 3 0 13 0 1 0 0 0 0 0 0 0 0
45 61 19 3 25 18 21 7 17 36 13 25 14 1 1 0 0 4 1 0
47 28 5 2 24 6 16 3 5 22 4 5 16 31 2 0 5 2 0 0
30 19 3 0 15 1 12 7 3 21 3 3 5 26 3 0 11 1 0 0
34 11 2 0 12 1 6 4 5 52 13 3 0 14 0 0 22 0 0 0
76 52 12 7 22 5 31 3 14 14 3 4 7 4 0 0 20 2 0 0
78 18 3 5 14 5 31 5 4 7 0 6 9 6 0 0 13 1 0 0
36 16 19 2 55 41 87 68 5 9 1 5 6 36 70 20 0 9 0 2
24 12 21 2 43 35 86 64 0 16 1 3 4 34 60 13 0 8 1 3
34 23 8 3 46 22 61 34 3 12 3 7 5 25 41 12 0 8 0 3
47 29 12 1 36 11 58 28 5 13 4 4 2 17 55 22 0 1 0 0
19 18 4 0 14 2 22 12 2 9 2 2 1 8 29 8 0 3 0 0
21 13 2 0 8 1 22 14 2 8 1 1 0 10 34 5 0 1 0 0
48 22 11 1 36 14 60 26 6 14 1 4 3 17 47 19 0 3 0 0
47 28 9 4 25 2 25 20 4 14 5 2 4 22 47 6 0 1 0 0
1 42 7 2 26 13 15 10 26 23 9 10 5 1 1 0 0 26 1 1
27 33 5 2 32 15 15 16 15 24 9 4 2 3 8 0 1 5 6 1
25 33 6 1 66 12 28 2 8 20 12 6 7 0 0 0 0 7 0 0
0 16 3 1 4 0 0 0 5 11 3 1 1 0 0 0 0 15 5 0
16 62 14 3 42 18 42 29 10 29 8 7 12 15 20 7 1 9 1 1
1 29 10 4 18 25 40 41 2 13 3 21 34 14 3 1 5 2 0 0
6 57 26 6 13 7 10 8 3 28 4 3 5 3 1 0 1 9 0 0

[1068]:
```
fig, axes = plt.subplots(N_subtypes, 1, figsize = (1, 12))

for idx, subtype in enumerate(subtypes):
    sample_type(subtype, axes[idx])
```

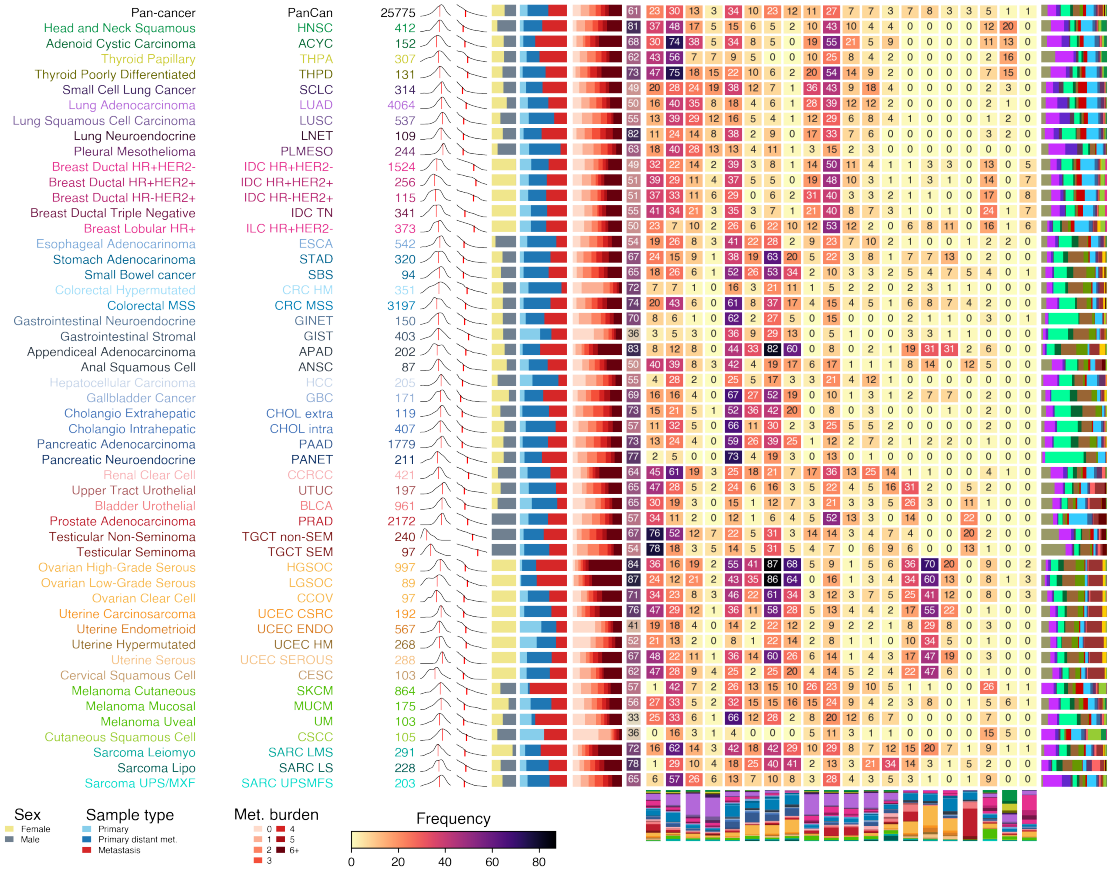
[887]:
```
fig, axes = plt.subplots(1, 20, figsize = (12, 2))

for idx, subtype in enumerate(met_count_cols[1:]):
    met_cancer_bar_plot(subtype, axes[idx])
```



1.2.2 Assembling everything

[1069]:
```
w_margin = 0.2
h_margin = 0.2
width_ratios = np.array([20, 2, 2, 1.5, 3, 3, 1, 20 + 19 * w_margin, 4])
height_ratios = np.array([4] + [1]*51)

total_width = width_ratios.sum() + w_margin * (width_ratios.shape[0] - 1)
total_height = height_ratios.sum() + h_margin * (height_ratios.shape[0] - 1)

xs = np.insert(np.cumsum(width_ratios[:-1]  + w_margin), 0, 0)/total_width
ys = np.insert(np.cumsum(height_ratios[:-1]  + h_margin), 0, 0)/total_height
ys = ys[::-1]
ws = width_ratios/total_width
hs = height_ratios/total_height
hs = hs[::-1]
w_margin = w_margin/total_width
h_margin = h_margin/total_height

fig = plt.figure(figsize = (total_width * 0.18, total_height * 0.14))

ax_text = [fig.add_axes([xs[0],ys[i],ws[0],hs[i]]) for i in range(N_subtypes)]
ax_age = [fig.add_axes([xs[1],ys[i],ws[1],hs[i]]) for i in range(N_subtypes)]
ax_os = [fig.add_axes([xs[2],ys[i],ws[2],hs[i]]) for i in range(N_subtypes)]
ax_sex = [fig.add_axes([xs[3],ys[i],ws[3],hs[i]]) for i in range(N_subtypes)]
ax_sample = [fig.add_axes([xs[4],ys[i],ws[4],hs[i]]) for i in range(N_subtypes)]
ax_burden = [fig.add_axes([xs[5],ys[i],ws[5],hs[i]]) for i in range(N_subtypes)]
ax_unspec_hm = fig.add_axes([xs[6], ys[-2], ws[6], 1 - ys[-2]])
ax_met_hm = fig.add_axes([xs[7], ys[-2], ws[7], 1 - ys[-2]])
ax_met_site_distr = [fig.add_axes([xs[8],ys[i],ws[8],hs[i]]) for i in␣
↪range(N_subtypes)]
```

```
ax_met_cancer_distr = [fig.add_axes([xs[7] + i * (ws[6] +␣
↪w_margin),ys[-1],ws[6],hs[-1]]) for i in range(len(met_count_cols[1:]))]

for idx, subtype in enumerate(subtypes):
    text_plot(subtype, ax_text[idx])
    age_plot(subtype, ax_age[idx])
    os_plot(subtype, ax_os[idx])
    sex_plot(subtype, ax_sex[idx])
    sample_type(subtype, ax_sample[idx])
    metastasis_burden(subtype, ax_burden[idx])
    met_site_bar_plot(subtype, ax_met_site_distr[idx])

unspec_met_heatmap_plot(ax_unspec_hm)
met_site_heatmap_plot(ax_met_hm)

for idx, met_subtype in enumerate(met_count_cols[1:]):
    met_cancer_bar_plot(met_subtype, ax_met_cancer_distr[idx])


plt.rcParams["legend.handlelength"] = 1
plt.rcParams["legend.labelspacing"] = 0.25
plt.rcParams["legend.frameon"] = False
plt.rcParams["legend.title_fontsize"] = 12.5
plt.rcParams["legend.loc"] = "upper left"

ax_sex[-1].legend(["Female", "Male"], title = "Sex", bbox_to_anchor = (-20,-1))
ax_sample[-1].legend(["Primary", "Primary distant met.", "Metastasis"], title =␣
↪"Sample type", bbox_to_anchor = (-9,-1))
ax_burden[-1].legend(["0", "1", "2", "3", "4", "5", "6+"], title = "Met.␣
↪burden", bbox_to_anchor = (-7,-1), ncol = 2, columnspacing = 0.5,␣
↪handletextpad = 0.5)


data = met_count_table.to_numpy()
data = (data / data[:, -1:] * 100).astype('int32')

norm = plt.Normalize(0, data[1:, 1:-1].max())
sm = plt.cm.ScalarMappable(cmap= sns.color_palette("magma_r", as_cmap=True),␣
↪norm=norm)
sm.set_array([])

cax = fig.add_axes([0.26, -0.008, 0.2, 0.02])
cbar = plt.colorbar(sm, cax=cax, orientation = 'horizontal')
cbar.ax.set_title('Frequency', fontsize = 12.5)

fig.savefig("fig1.png", bbox_inches = "tight")
```



1.3 Comparing FGA, WGD, and TMB values

[985]:
```
# Processing our ASCETS-calculated WGD values
our_wgd = pd.read_csv("../GIIa3_github/Genomics-II-Group/Plot_2/Figure2B/
 ↪tmb_high_and_wgd.csv")
our_wgd = our_wgd.loc[:, ["SUBTYPE", "SAMPLE_TYPE", "WGD"]]

## change subtype name from curated_subtype_display to curated_subtype
our_wgd = our_wgd.rename(columns = {"SUBTYPE": "curated_subtype_display"}).
 ↪set_index("curated_subtype_display")
display_names = table1.loc[:, ["curated_subtype", "curated_subtype_display"]].
 ↪set_index("curated_subtype_display")
our_wgd = our_wgd.join(display_names, on = "curated_subtype_display").
 ↪set_index("curated_subtype")

## pivoting table for easier comparison
our_wgd = our_wgd.pivot(columns = ["SAMPLE_TYPE"])
our_wgd.columns = our_wgd.columns.map("_".join)
```

```
our_wgd = our_wgd.rename(columns = {"WGD_Metastasis":"our_metastasis_pc",␣
↪"WGD_Primary":"our_primary_pc"})

# Importing the paper's WGD values
true_wgd = pd.read_csv("table2_1.csv")
true_wgd = true_wgd[true_wgd['alteration'] == 'WGD'].loc[:, ["tumor_type",␣
↪"metastasis_pc", "primary_pc"]]

## converting percentages from strings to floats
true_wgd['primary_pc'] = true_wgd['primary_pc'].str.rstrip('%').astype('float')␣
↪/ 100.0
true_wgd['metastasis_pc'] = true_wgd['metastasis_pc'].str.rstrip('%').
↪astype('float') / 100.0
true_wgd = true_wgd.rename(columns = {"tumor_type":"curated_subtype"}).
↪set_index("curated_subtype")

# Join both tables
all_wgd = true_wgd.join(our_wgd, on = "curated_subtype")
```

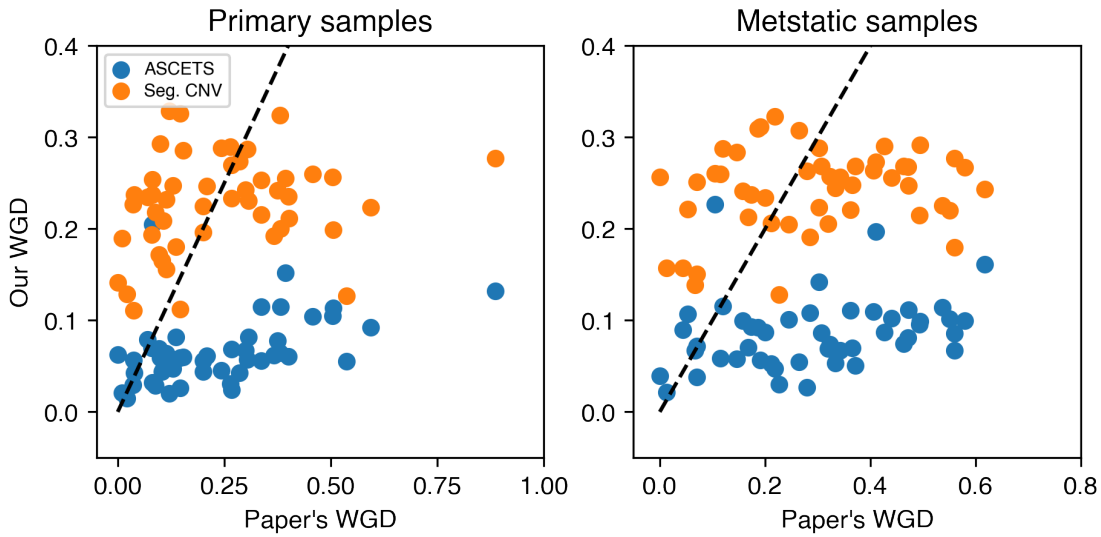
1.3.1 Computing WGDs from segmented CNV data

[995]:
```
# Importing all segmented CNV data
cnv = pd.read_table("../GIIa3_github/Genomics-II-Group/Data/data_cna_hg19.seg",␣
↪sep = "\t")
cnv = cnv.rename(columns = {"ID":"sample_id"})

# Computing WGDs
## removing sex chromosomes
cnv = cnv.loc[~cnv["chrom"].isin(["X", "Y"])]

## computing genome length fractions
cnv['length'] = cnv["loc.end"] - cnv["loc.start"]
all_lengths = cnv.groupby("sample_id").agg({"length": np.sum})
cnv = cnv.join(all_lengths, on="sample_id", rsuffix = "_total")
cnv["length_fraction"] = cnv["length"]/cnv["length_total"]

## adding curated_subtype, sample_type, and tumor_purity information
extra_info = table2.loc[:,["sample_id", "sample_type", "curated_subtype",␣
↪"tumor_purity"]].set_index("sample_id")
cnv = cnv.join(extra_info, on = "sample_id")

## computing sample-wise average ratios
cnv['ratio'] = 2**cnv["seg.mean"]
ratio_avgs = cnv.groupby("sample_id").agg({"ratio": np.median})
cnv = cnv.join(ratio_avgs, on="sample_id", rsuffix = "_avg")
```

```
## centering sample-wise CN ratios
cnv['ratio'] = cnv['ratio'] - cnv["ratio_avg"] + 1

## cleaning up purity values
cnv["tumor_purity"] = cnv["tumor_purity"]/100
cnv.loc[cnv["tumor_purity"] == 0, "tumor_purity"] = 0.1
cnv.loc[cnv["tumor_purity"].isna(), "tumor_purity"] = 0.5

## computing purity-adjusted ratios
cnv["adj_ratio"] = (cnv["ratio"] + cnv["tumor_purity"] - 1)/cnv["tumor_purity"]

## cleaning up negative ratios
cnv.loc[cnv["adj_ratio"]<=0, "adj_ratio"] = 0.01

## filtering fractions based on copy number ratio
cnv["filtered_fractions"] = cnv["length_fraction"] * (cnv["adj_ratio"]>=0) * np.
↪minimum(np.log2(cnv["adj_ratio"]), 1)**cnv["tumor_purity"]

## grouping, aggregation, table pivoting
cnv = cnv.groupby(["curated_subtype", "sample_type"]).agg({"filtered_fractions":
↪ np.sum})
cnv = cnv.reset_index(level=1).pivot(columns = ["sample_type"])
cnv.columns = cnv.columns.map('_'.join)
cnv = cnv.rename(columns = {"filtered_fractions_Metastasis":
↪"our_metastasis_pc2",
                            "filtered_fractions_Primary":"our_primary_pc2"})

## getting sample sizes and calculating percentages
sample_sizes = table1.loc[:, ["curated_subtype", "primary_n", "metastasis_n"]].
↪set_index("curated_subtype")
cnv = cnv.join(sample_sizes, on = "curated_subtype")
cnv["our_metastasis_pc2"] = cnv["our_metastasis_pc2"]/cnv["metastasis_n"]
cnv["our_primary_pc2"] = cnv["our_primary_pc2"]/cnv["primary_n"]

# Adding to all_wgd table for comparison
# all_wgd = all_wgd.join(cnv, on = "curated_subtype")
```

[1070]:
```
fig, axes = plt.subplots(1,2, figsize = (7,3))

ax = axes[0]
ax.scatter(all_wgd["primary_pc"], all_wgd["our_primary_pc"])
ax.scatter(all_wgd["primary_pc"], cnv["our_primary_pc2"])
ax.legend(["ASCETS", "Seg. CNV"], frameon = True)

ax.plot([0,1], [0,1], c = "k", linestyle = "dashed")
ax.set_xlabel("Paper's WGD")
ax.set_ylabel("Our WGD")
```

```
ax.set_xlim([-0.05, 1])
ax.set_ylim([-0.05, 0.4])
ax.set_title("Primary samples")

ax = axes[1]
ax.scatter(all_wgd["metastasis_pc"], all_wgd["our_metastasis_pc"])
ax.scatter(all_wgd["metastasis_pc"], cnv["our_metastasis_pc2"])
ax.plot([0,1], [0,1], c = "k", linestyle = "dashed")
ax.set_xlabel("Paper's WGD")
ax.set_xlim([-0.05, 0.8])
ax.set_ylim([-0.05, 0.4])
ax.set_title("Metstatic samples");
```

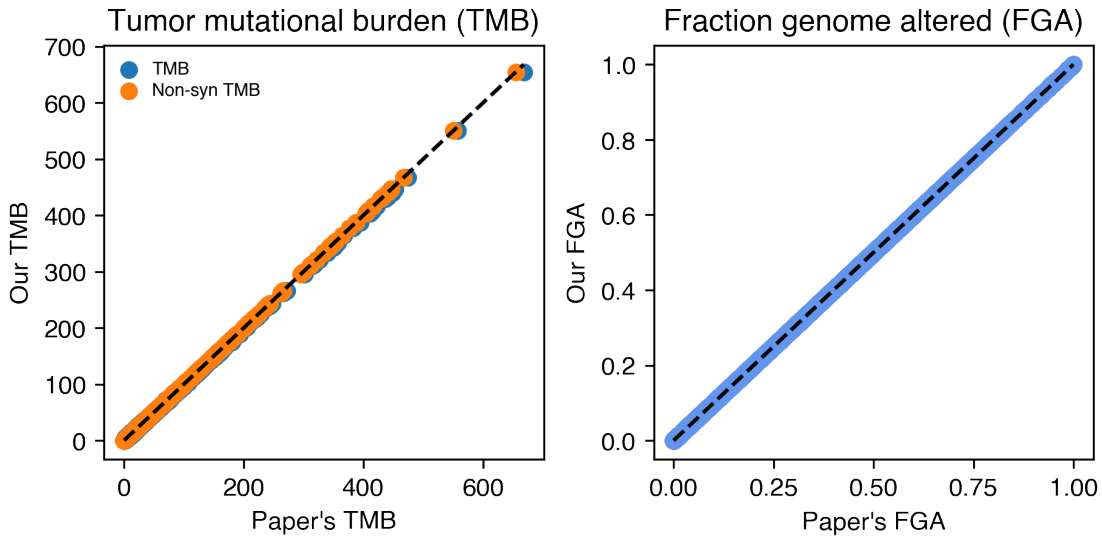


1.3.2 Comparing FGA and TMB values

[1001]:
```
# Importing our TMB and FGA values
tmb_fga_table = pd.read_csv("../GIIa3_github/Genomics-II-Group/Plot_2/FGA/
 ↪calculated_TMB_and_FGA.csv")
```

[1002]:
```
fig, axes = plt.subplots(1,2,figsize = (7,3))

ax = axes[0]
ax.scatter(tmb_fga_table["Sup_TMB"], tmb_fga_table["Our_TMB"])
ax.scatter(tmb_fga_table["TMB_NONSYNONYMOUS"], tmb_fga_table["Our_TMB"])
ax.legend(["TMB", "Non-syn TMB"])
x_max = tmb_fga_table["Sup_TMB"].max()
ax.plot([0, x_max], [0, x_max], c = "k", linestyle = "dashed")
ax.set_title("Tumor mutational burden (TMB)")
```

```
ax.set_xlabel("Paper's TMB")
ax.set_ylabel("Our TMB")

ax = axes[1]
ax.scatter(tmb_fga_table["Sup_FGA"], tmb_fga_table["Our_FGA"], color =␣
↪"cornflowerblue")
x_max = tmb_fga_table["Sup_FGA"].max()
ax.plot([0, x_max], [0, x_max], c = "k", linestyle = "dashed")
ax.set_title("Fraction genome altered (FGA)")
ax.set_xlabel("Paper's FGA")
ax.set_ylabel("Our FGA")

fig.subplots_adjust(wspace = 0.25)
```



1.4 Figure 2B

[1003]:
```
# Assembling the table
fig2b_table = table1.loc[:, ["curated_subtype",
                "curated_subtype_display",
                "total_n", "primary_n", "organ_system",
                "metastasis_n", "color_subtype"]].copy()

fig2b_table = fig2b_table.set_index("curated_subtype", drop = False)
fig2b_table.index.names = ["subtype"]

# adding wgd
fig2b_table = fig2b_table.join(cnv.loc[:, ["our_metastasis_pc2",␣
↪"our_primary_pc2"]], on = "curated_subtype")
```

```
# adding fga medians
tmb_fga_table["Our_FGA"] = tmb_fga_table["Our_FGA"]*100
tmb_fga_table = tmb_fga_table.rename(columns={"SUBTYPE":
 ↪"curated_subtype_display"})
tmb_fga_table = tmb_fga_table.join(display_names, on="curated_subtype_display")
medians = tmb_fga_table.loc[tmb_fga_table["SAMPLE_TYPE"]=="Metastasis"]\
                        .groupby("curated_subtype").
 ↪agg(fga_median=("Our_FGA",np.median))
fig2b_table = fig2b_table.join(medians, on = "curated_subtype")
median_max = fig2b_table.groupby("organ_system").agg(fga_median_max =␣
 ↪("fga_median", np.max))
fig2b_table = fig2b_table.join(median_max, on = "organ_system")
fig2b_table = fig2b_table.sort_values(by = ["fga_median_max", "fga_median"],␣
 ↪ascending = False)

# adding fraction of samples with high tmb
tmb_fga_table["TMB_hi"] = tmb_fga_table["Our_TMB"]>10
num_hi_tmb = tmb_fga_table.groupby(["curated_subtype", "SAMPLE_TYPE"]) \
                          .agg(num_hi_TMB=("TMB_hi", np.sum)) \
                          .reset_index(1).pivot(columns = "SAMPLE_TYPE")
num_hi_tmb.columns = num_hi_tmb.columns.map('_'.join)

fig2b_table = fig2b_table.join(num_hi_tmb, on = "curated_subtype")
fig2b_table["num_hi_TMB_Metastasis"] = fig2b_table["num_hi_TMB_Metastasis"]/
 ↪fig2b_table["metastasis_n"]
fig2b_table["num_hi_TMB_Primary"] = fig2b_table["num_hi_TMB_Primary"]/
 ↪fig2b_table["primary_n"]
```

OncoKB Actionability

[1004]:
```
oncokb_mut = pd.read_table("../GIIa3_github/Genomics-II-Group/Plot_2/OncoKB/
 ↪oncokb_annotated_mutations.txt", sep = "\t")
oncokb_fus = pd.read_table("../GIIa3_github/Genomics-II-Group/Plot_2/OncoKB/
 ↪oncokb_annotated_fusions.txt", sep = "\t")
oncokb_cna = pd.read_table("../GIIa3_github/Genomics-II-Group/Plot_2/OncoKB/
 ↪oncokb_annotated_cna.txt", sep = "\t")

oncokb_mut = oncokb_mut[oncokb_mut["VARIANT_IN_ONCOKB"]==True]
oncokb_fus = oncokb_fus[oncokb_fus["VARIANT_IN_ONCOKB"]==True]
oncokb_cna = oncokb_cna[oncokb_cna["VARIANT_IN_ONCOKB"]==True]

subtype_info = table2.loc[:, ["sample_id", "curated_subtype", "sample_type"]]

oncokb_mut = oncokb_mut.merge(subtype_info, left_on="Tumor_Sample_Barcode",␣
 ↪right_on="sample_id")
```

```
oncokb_fus = oncokb_fus.merge(subtype_info, left_on="Tumor_Sample_Barcode",␣
↪right_on="sample_id")
oncokb_cna = oncokb_cna.merge(subtype_info, left_on="SAMPLE_ID",␣
↪right_on="sample_id")
```

```
/usr/local/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3337:
DtypeWarning: Columns (21) have mixed types.Specify dtype option on import or
set low_memory=False.
  if (await self.run_code(code, result,  async_=asy)):
/usr/local/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3337:
DtypeWarning: Columns (9,10,11,12,13,15,17,18) have mixed types.Specify dtype
option on import or set low_memory=False.
  if (await self.run_code(code, result,  async_=asy)):
```

[1005]:
```
# count mutations once per subtype/sample type combination (?)
oncokb_mut = oncokb_mut.drop_duplicates(["curated_subtype", "sample_type",␣
↪"Hugo_Symbol", "HGVSp_Short"])
oncokb_fus = oncokb_fus.drop_duplicates(["curated_subtype", "sample_type",␣
↪"Hugo_Symbol", "Fusion"])
oncokb_cna = oncokb_cna.drop_duplicates(["curated_subtype", "sample_type",␣
↪"HUGO_SYMBOL", "ALTERATION"])
```

[1006]:
```
oncokb_mut = oncokb_mut.groupby(["curated_subtype", "sample_type"]).agg(
                                        Oncogenic=("ONCOGENIC", lambda x: x.
↪str.match("Oncogenic").sum()),
                                        L1=("LEVEL_1", lambda x: x.notna().
↪sum()),
                                        L2=("LEVEL_2", lambda x: x.notna().
↪sum()),
                                        L3A=("LEVEL_3A", lambda x: x.notna().
↪sum()),
                                        L3B=("LEVEL_3B", lambda x: x.notna().
↪sum()),
                                        L4=("LEVEL_4", lambda x: x.notna().
↪sum()),
                                        VUS=("ONCOGENIC", lambda x: x.
↪isin(["Inconclusive", "Unknown"]).sum()),
                                        NA=("HIGHEST_LEVEL", lambda x: x.
↪isna().sum()))
oncokb_fus = oncokb_fus.groupby(["curated_subtype", "sample_type"]).agg(
                                        Oncogenic=("ONCOGENIC", lambda x: x.
↪str.match("Oncogenic").sum()),
                                        L1=("LEVEL_1", lambda x: x.notna().
↪sum()),
                                        L2=("LEVEL_2", lambda x: x.notna().
↪sum()),
```

```
                                        L3A=("LEVEL_3A", lambda x: x.notna().
↪sum()),
                                        L3B=("LEVEL_3B", lambda x: x.notna().
↪sum()),
                                        L4=("LEVEL_4", lambda x: x.notna().
↪sum()),
                                        VUS=("ONCOGENIC", lambda x: x.
↪isin(["Inconclusive", "Unknown"]).sum()),
                                        NA=("HIGHEST_LEVEL", lambda x: x.
↪isna().sum()))
oncokb_cna = oncokb_cna.groupby(["curated_subtype", "sample_type"]).agg(
                                        Oncogenic=("ONCOGENIC", lambda x: x.
↪str.match("Oncogenic").sum()),
                                        L1=("LEVEL_1", lambda x: x.notna().
↪sum()),
                                        L2=("LEVEL_2", lambda x: x.notna().
↪sum()),
                                        L3A=("LEVEL_3A", lambda x: x.notna().
↪sum()),
                                        L3B=("LEVEL_3B", lambda x: x.notna().
↪sum()),
                                        L4=("LEVEL_4", lambda x: x.notna().
↪sum()),
                                        VUS=("ONCOGENIC", lambda x: x.
↪isin(["Inconclusive", "Unknown"]).sum()),
                                        NA=("HIGHEST_LEVEL", lambda x: x.
↪isna().sum()))
```

[1007]:
```
OncoKB = oncokb_mut.merge(oncokb_cna, left_index = True, right_index = True,␣
↪suffixes =("_mut", "_cna"))
OncoKB = OncoKB.merge(oncokb_fus, left_index = True, right_index = True)

OncoKB["Oncogenic_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("Onco")].
↪sum(1)
OncoKB["L1_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("L1")].sum(1)
OncoKB["L2_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("L2")].sum(1)
OncoKB["L3A_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("L3A")].sum(1)
OncoKB["L4_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("L4")].sum(1)
OncoKB["VUS_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("VUS")].sum(1)
OncoKB["None_total"] = OncoKB.loc[:, OncoKB.columns.str.contains("None")].sum(1)

OncoKB = OncoKB.loc[:, OncoKB.columns.str.contains("_total")]
OncoKB = OncoKB.div(OncoKB.sum(1), axis = 0)
```

[1008]:
```
OncoKB = OncoKB.reset_index(1).pivot(columns = "sample_type")
```

```
OncoKB = OncoKB.reindex(subtypes[1:])
OncoKB[OncoKB.isna()] = 0

OncoKB.loc[:, ("None_total", "Metastasis")] = np.round(1 - OncoKB.
↪swaplevel(axis = 1).loc[:, "Metastasis"].sum(1))
OncoKB.loc[:, ("None_total", "Primary")] = np.round(1 - OncoKB.swaplevel(axis =␣
↪1).loc[:, "Primary"].sum(1))
```

[1009]:
```
def actionability_plot(y_positions, y_lim, ax, h=0.4, p=0.04):
    ax.set_title("Actionability", fontsize = 7)
    actions = ['L1_total', 'L2_total', 'L3A_total',
               'L4_total','Oncogenic_total', 'VUS_total', 'None_total']
    colors = ["tab:green", "tab:blue", "tab:purple",
               "tab:pink", "darkslategray", "blanchedalmond", "lightgrey",␣
↪"white"]

    prim_left, meta_left = 0, 0
    for idx, action in enumerate(actions):
        ax.barh(y = y_positions+h/2+p/2, height = h, left = prim_left,
                            width = OncoKB[(action, "Primary")],
                            color = colors[idx])
        ax.barh(y = y_positions-h/2-p/2, height = h, left = meta_left,
                            width = OncoKB[(action, "Metastasis")],
                            color = colors[idx],
                            label = "_nolegend_")
        prim_left = prim_left + OncoKB[(action, "Primary")]
        meta_left = meta_left + OncoKB[(action, "Metastasis")]

    ax.set_ylim(y_lim)
    ax.set_yticks([])
    ax.spines['right'].set_visible(False)
    ax.set_xticks([0, 1])
    ax.set_xticklabels([0, 1], fontsize = 7)
    ax.set_xlim([0, 1])
    ax.set_xlabel("Fraction", fontsize = 7);
```

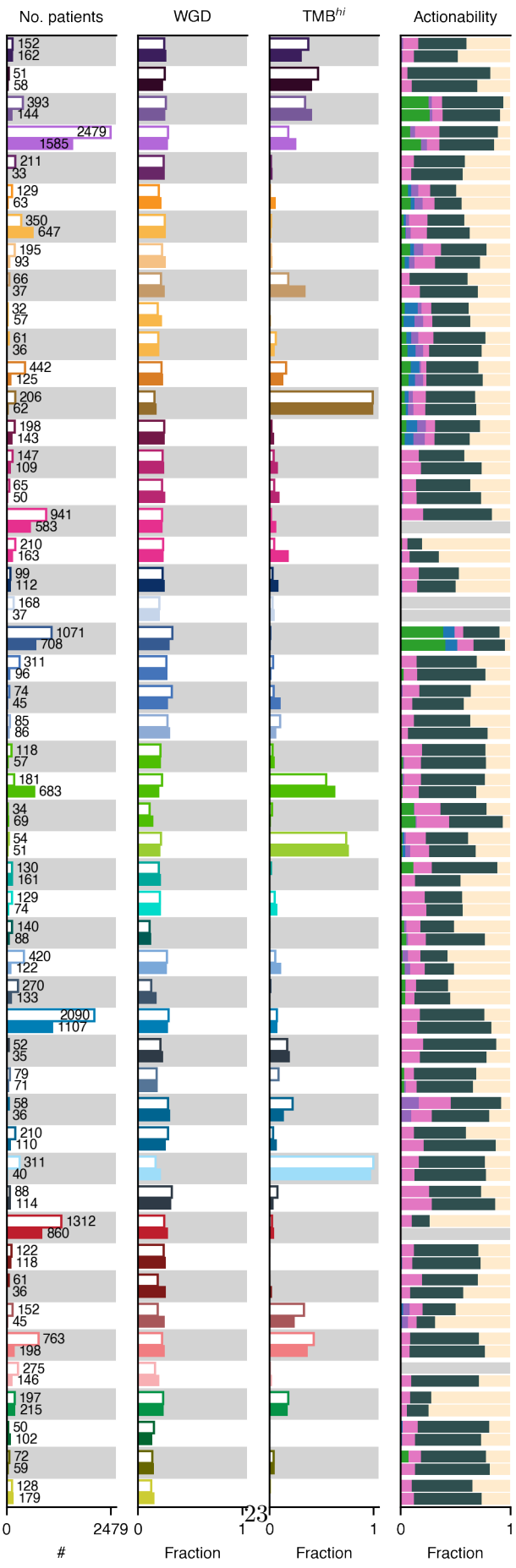
[1010]:
```
def text_plot(y_positions, y_lim, ax):
    subtypes = fig2b_table.index.to_list()
    for subtype, y_pos in zip(subtypes, y_positions):
        title = fig2b_table.loc[subtype, "curated_subtype_display"]
        sample_size = fig2b_table.loc[subtype, "total_n"]
        text = f"{title} ({str(sample_size)})"
        ax.text(1, y_pos, text,
                ha = "right", va = "center",
                color = fig2b_table.loc[subtype, "color_subtype"])
    ax.set_ylim(y_lim)
    ax.set_xlim([0, 1])
```

```
    ax.axis('off')
```

[1011]:
```
def two_bars_plot(prim_value, meta_value, annot, y_positions, y_lim, ax,␣
↪xlabel, x_max = False, h=0.4, title = ""):
    ax.set_title(title, fontsize = 7)
    # grey areas in background
    for idx, y_pos in enumerate(y_positions):
        if idx % 2 == 0:
            ax.axhspan(y_pos-0.5, y_pos+0.5, color = "lightgrey")

    # draw bars
    colors = fig2b_table["color_subtype"]
    subtypes = fig2b_table.index.to_list()

    prim_bars = ax.barh(y = y_centers+h/2, height = h,
                        width = fig2b_table[prim_value],
                        color = "white", edgecolor = colors)
    meta_bars = ax.barh(y =  y_centers-h/2, height = h,
                        width = fig2b_table[meta_value],
                        color = colors)

    if annot:
        for subtype,y_pos in zip(subtypes, y_positions):
            prim_x = fig2b_table.loc[subtype, "primary_n"]
            meta_x = fig2b_table.loc[subtype, "metastasis_n"]
            ax.text(prim_x-100 if prim_x>1400 else prim_x+100,
                    y_pos+h/2, str(prim_x),
                    ha = "right" if prim_x>1400 else "left",
                    va = "center", fontsize = 6)
            ax.text(meta_x-100 if meta_x>1400 else meta_x+100,
                    y_pos-h/2, str(meta_x),
                    ha = "right" if meta_x>1400 else "left",
                    va = "center", fontsize = 6)

    ax.set_ylim(y_lim)
    ax.set_yticks([])
    ax.spines['right'].set_visible(False)

    if not x_max:
        x_max = fig2b_table.loc[:, [prim_value, meta_value]].max().max()
    ax.set_xticks([0, x_max])
    ax.set_xticklabels([0, x_max], fontsize = 7)
    ax.set_xlim([0, 1.05*x_max])
    ax.set_xlabel(xlabel, fontsize = 7);
```

[1012]:
```
fig, ax = plt.subplots(1,4, figsize = (4, 12))
```

```
two_bars_plot(prim_value = "primary_n",
              meta_value = "metastasis_n",
              annot=True,
              y_positions = np.arange(50, 0, -1),
              y_lim = [0.5, 50.5],
              ax = ax[0],
              xlabel = "#",
              h=0.4,
              title = "No. patients")

two_bars_plot(prim_value = "our_primary_pc2",
              meta_value = "our_metastasis_pc2",
              annot = False,
              y_positions = np.arange(50, 0, -1),
              y_lim = [0.5, 50.5],
              ax = ax[1],
              xlabel = "Fraction",
              x_max = 1,
              h=0.4,
              title = "WGD")

two_bars_plot(prim_value = "num_hi_TMB_Primary",
              meta_value = "num_hi_TMB_Metastasis",
              annot = False,
              y_positions = np.arange(50, 0, -1),
              y_lim = [0.5, 50.5],
              ax = ax[2],
              xlabel = "Fraction",
              x_max = 1,
              h=0.4,
              title = r"TMB$^{hi}$")

actionability_plot(y_positions=np.arange(50, 0, -1),
                   y_lim = [0.5, 50.5],
                   ax=ax[3], h=0.4, p=0.04)
```


No. patients
WGD
TMB^hi
Actionability
152
162
51
58
393
144
2479
1585
211
33
129
63
350
647
195
93
66
37
32
57
61
36
442
125
206
62
198
143
147
109
65
50
941
583
210
163
99
112
168
37
1071
708
311
96
74
45
85
86
118
57
181
683
34
69
54
51
130
161
129
74
140
88
420
122
270
133
2090
1107
52
35
79
71
58
36
210
110
311
40
88
114
1312
860
122
118
61
36
152
45
763
198
275
146
197
215
50
102
72
59
128
179
0
2479
#
0
1
Fraction
0
1
Fraction
0
1
Fraction

1.4.1 FGA violin plots

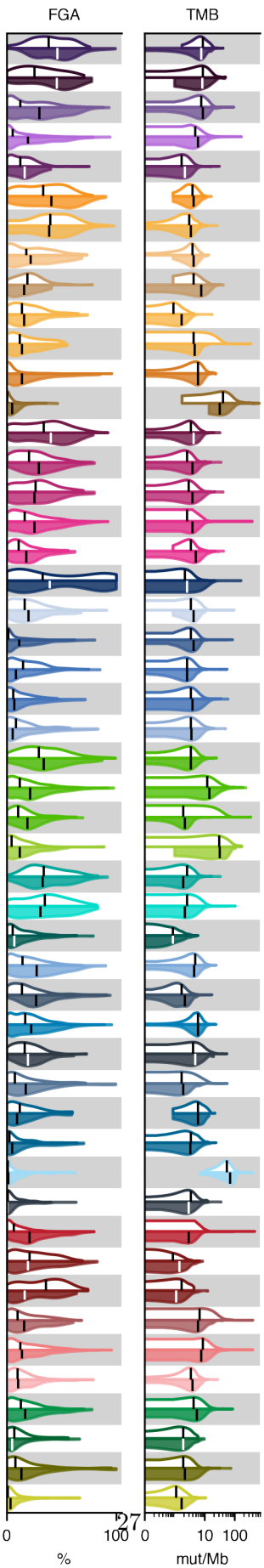
[1013]:
```
fga_data = tmb_fga_table.groupby(["curated_subtype", "SAMPLE_TYPE"])["Our_FGA"].
↪apply(pd.Series.to_list).to_dict()
tmb_data = tmb_fga_table.groupby(["curated_subtype", "SAMPLE_TYPE"])["Our_TMB"].
↪apply(pd.Series.to_list).to_dict()

# adjust color contrasts for median lines
cols = [h.strip("#") for h in fig2b_table['color_subtype']]
cols_rgb = np.array([[int(h[i:i+2], 16) for i in (0,2,4)] for h in cols])
fig2b_table["median_line_cols"] = ['white' if x else 'black' for x in (cols_rgb␣
↪@ np.array([[0.299],[0.587],[0.114]]))<70]
```

[1014]:
```
def violin_plot(data, y_positions, y_lim, ax, xlabel, x_max=False, logscale =␣
↪False, title = ""):
    ax.set_title(title, fontsize = 7)
    ax.set_yticks([])
    ax.spines['right'].set_visible(False)
    if x_max:
        ax.set_xlim([0, 1.05*x_max])
        ax.set_xticks([0, x_max])
        ax.set_xticklabels([0, x_max], fontsize = 7)
    ax.set_xlabel(xlabel, fontsize = 7)
    ax.set_ylim(y_lim)
    if logscale:
        ax.set_xscale('log')
        ax.set_xlim([0.1, 700])
        ax.set_xticks([0.1, 10, 100])
        ax.set_xticklabels([0, 10, 100], fontsize = 7)

    # grey areas in background
    for idx, y_pos in enumerate(y_positions):
        if idx % 2 == 0:
            ax.axhspan(y_pos-0.5, y_pos+0.5, color = "lightgrey")

    # get data
    subtypes = fig2b_table.index.to_list()
    prim_data = [data[(st, "Primary")] for st in subtypes]
    meta_data = [data[(st, "Metastasis")] for st in subtypes]

    # draw violin plots
    prim_vp = ax.violinplot(prim_data, y_positions,
                            vert = False, showextrema=False, showmeans=False,␣
↪showmedians=True, widths = 0.8);
```

```
   meta_vp = ax.violinplot(meta_data, y_positions,
                           vert = False, showextrema=False, showmeans=False,␣
↪showmedians=True, widths = 0.8);

   # fix median lines width and color
   prim_vp['cmedians'].set_linewidths(1)
   prim_vp['cmedians'].set_colors("k")
   meta_vp['cmedians'].set_linewidths(1)
   meta_vp['cmedians'].set_colors(fig2b_table["median_line_cols"])


   for idx, (prim_b, meta_b, st) in enumerate(zip(prim_vp['bodies'],␣
↪meta_vp['bodies'], subtypes)):
       # fix colors and line widths
       type_col = fig2b_table.loc[st, "color_subtype"]

       prim_b.set_edgecolor(type_col)
       prim_b.set_facecolor("white")
       prim_b.set_linewidths(1.25)
       prim_b.set_alpha(1)

       meta_b.set_edgecolor(type_col)
       meta_b.set_facecolor(type_col)
       meta_b.set_linewidths(1.25)
       meta_b.set_alpha(0.8)

       # split violins in half and fix median lines
       # primary data
       ## get mean y position
       path = prim_b.get_paths()[0].vertices
       m = np.mean(path[:, 1])
       # modify the paths to not go below mean y position
       prim_b.get_paths()[0].vertices[:, 1] = np.clip(path[:, 1], m, +np.inf)

       ## fix the median lines
       mverts = prim_vp["cmedians"].get_paths()[idx].vertices
       x_idx = np.argsort(np.abs(path[:, 0] - mverts[0,0]))[:3]
       mverts[0, 1] = path[x_idx, 1].min()
       mverts[1,1] = path[x_idx, 1].max()
       prim_vp["cmedians"].get_paths()[idx].vertices = mverts

       # metastasis data
       path = meta_b.get_paths()[0].vertices
       m = np.mean(path[:, 1])
       meta_b.get_paths()[0].vertices[:, 1] = np.clip(path[:, 1], -np.inf, m)

       mverts = meta_vp["cmedians"].get_paths()[idx].vertices
```

```
        x_idx = np.argsort(np.abs(path[:, 0] - mverts[0,0]))[:3]
        mverts[0, 1] = path[x_idx, 1].min()
        mverts[1,1] = path[x_idx, 1].max()
        meta_vp["cmedians"].get_paths()[idx].vertices = mverts
```

[1015]:
```
fig, ax = plt.subplots(1,2,figsize = (2, 12))

violin_plot(data = fga_data,
            y_positions = np.arange(50, 0, -1),
            y_lim = [0.5,50.5],
            ax = ax[0],
            xlabel = "%",
            x_max= 100,
            title = "FGA")

violin_plot(data = tmb_data,
            y_positions = np.arange(50, 0, -1),
            y_lim = [0.5,50.5],
            ax = ax[1],
            xlabel = "mut/Mb",
            logscale=True,
            title = "TMB")
```

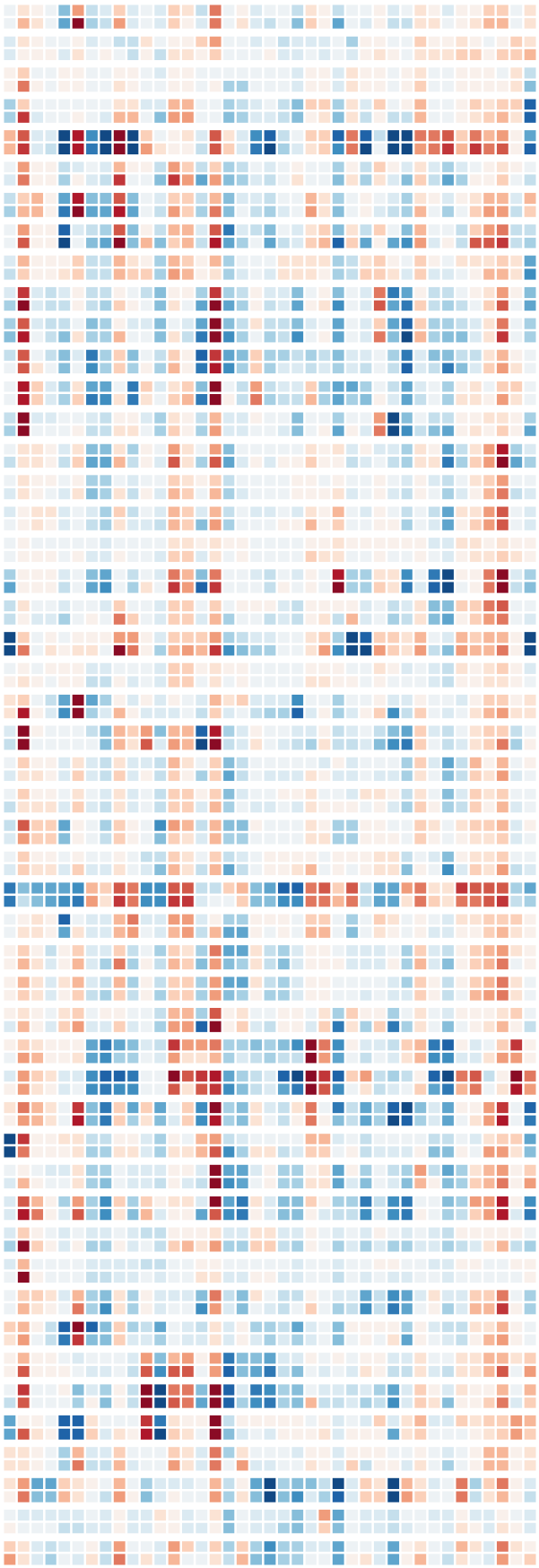

FGA
TMB
0
100
%
0
10
100
mut/Mb

1.4.2 Arm-level CNAs

[1016]:
```
cna = pd.read_csv("../GIIa3_github/Genomics-II-Group/Plot_2/aSCNAs/
 ↪subtype_mean_arm_level_cna.csv")
cna = cna.merge(display_names, left_on="SUBTYPE", right_index = True).
 ↪set_index(["curated_subtype", "SAMPLE_TYPE"])

chrom_arms = ['X1p', 'X1q',
              'X2p', 'X2q',
              'X3p', 'X3q',
              'X4p', 'X4q',
              'X5p', 'X5q',
              'X6p', 'X6q',
              'X7p', 'X7q',
              'X8p', 'X8q',
              'X9p', 'X9q',
              'X10p', 'X10q',
              'X11p', 'X11q',
              'X12p', 'X12q',
              'X13q',
              'X14q',
              'X15q',
              'X16p', 'X16q',
              'X17p', 'X17q',
              'X18p', 'X18q',
              'X19p', 'X19q',
              'X20p', 'X20q',
              'X21q',
              'X22q']
```

[1017]:
```
def cna_heatmap_plot(subtype, ax):
    data = cna.loc[[(subtype, "Primary"), (subtype, "Metastasis")], chrom_arms].
 ↪to_numpy()
    sns.heatmap(data, cmap=sns.color_palette("RdBu_r", 20),
            center = 0, lw = 0.5, ax = ax, cbar = False,
            vmin = -0.26, vmax = 0.26)
    ax.axis('off');
```

[1071]:
```
fig, axes = plt.subplots(50,1, figsize = (4, 12))

for ax, st in zip(axes, subtypes[1:]):
    cna_heatmap_plot(st, ax)
```



1.4.3 Assembling Figure 2B

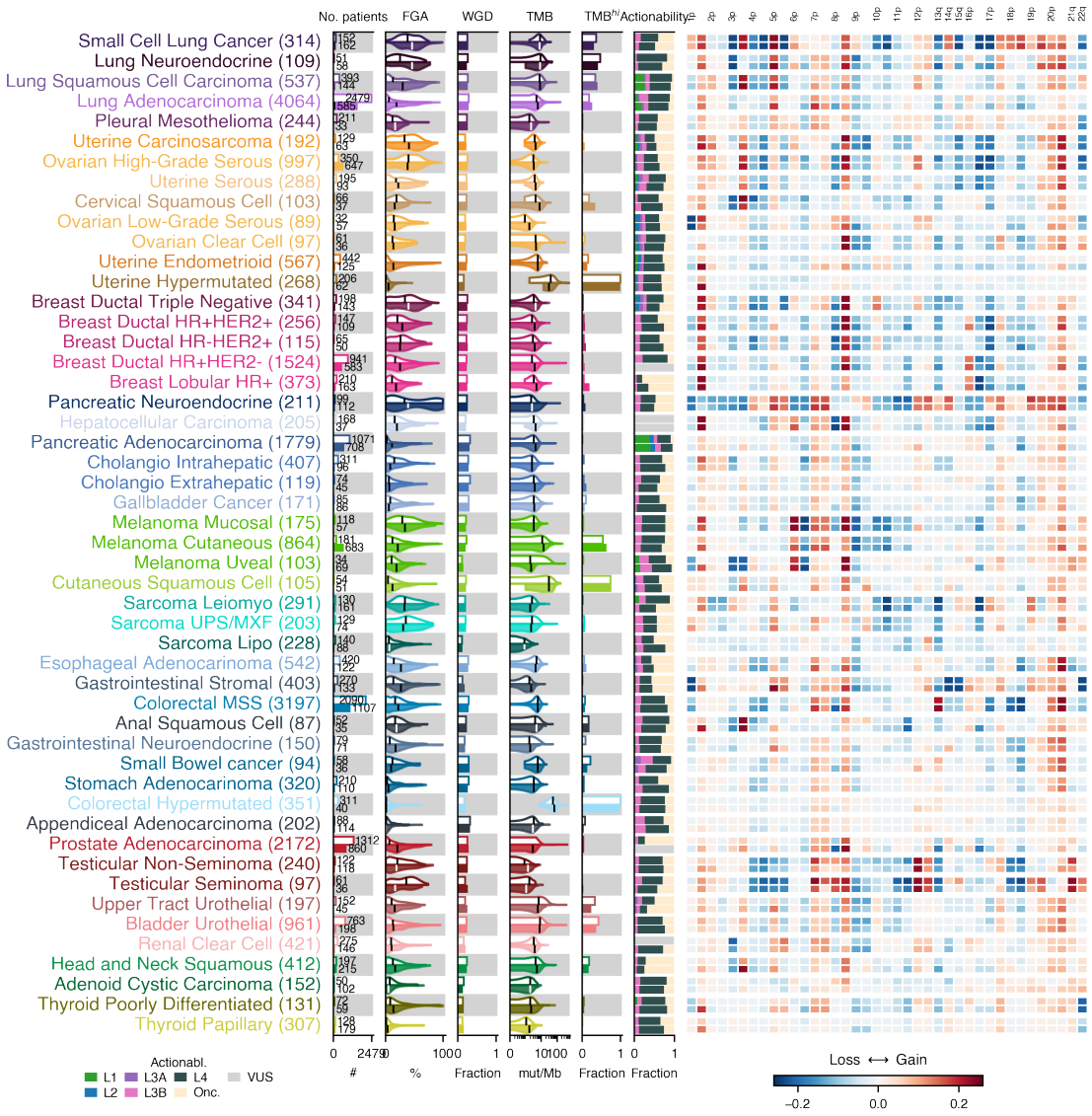
[1034]:
```
chrom_num = [x[1:] for x in chrom_arms]
```

[1036]:
```
heatmap_top = ['1p', '', '2p', '', '3p', '', '4p', '', '5p', '', '6p', '',␣
↪'7p', '',
               '8p', '', '9p', '', '10p', '', '11p', '', '12p', '', '13q',␣
↪'14q',
               '15q', '16p', '', '17p', '', '18p', '', '19p', '', '20p', '',␣
↪'21q', '22q']
```

[1057]:
```
w_margin = 0.3
pad_b = 0.1
pad_t = 0.1
width_ratios = np.array([7,1,1.5,1,1.5,1,1,10])
height_ratios = np.array([1]*50)

total_width = width_ratios.sum() + w_margin * (len(width_ratios) - 1)
total_height = height_ratios.sum()

xs = np.insert(np.cumsum(width_ratios[:-1]  + w_margin), 0, 0)/total_width
ys = np.insert(np.cumsum(height_ratios[:-1]), 0, 0)/total_height
ys = ys[::-1]
ws = width_ratios/total_width
hs = (1 - pad_b - pad_t)/total_height
w_margin = w_margin/total_width
pad_b = pad_b/total_height
fig = plt.figure(figsize = (total_width * 0.3, total_height * 0.15))

ax_text = fig.add_axes([xs[0], ys[-1], ws[0], 1])
ax_no_samples = fig.add_axes([xs[1], ys[-1], ws[1], 1])
ax_fga = fig.add_axes([xs[2], ys[-1], ws[2], 1])
ax_wgd = fig.add_axes([xs[3], ys[-1], ws[3], 1])
ax_tmb = fig.add_axes([xs[4], ys[-1], ws[4], 1])
ax_tmb_hi = fig.add_axes([xs[5], ys[-1], ws[5], 1])
ax_action = fig.add_axes([xs[6], ys[-1], ws[6], 1])
ax_heatmaps = [fig.add_axes([xs[7], y+pad_b, ws[7], hs]) for y in ys]

# plot everything

y_positions = np.arange(50, 0, -1)
y_lim = [0.5, 50.5]

text_plot(y_positions, y_lim, ax = ax_text)
```

```
two_bars_plot(prim_value = "primary_n",
              meta_value = "metastasis_n",
              annot=True,
              y_positions = y_positions,
              y_lim = y_lim,
              ax = ax_no_samples,
              xlabel = "#",
              h=0.4,
              title = "No. patients")

two_bars_plot(prim_value = "our_primary_pc2",
              meta_value = "our_metastasis_pc2",
              annot = False,
              y_positions = y_positions,
              y_lim = y_lim,
              ax = ax_wgd,
              xlabel = "Fraction",
              x_max = 1,
              h=0.4,
              title = "WGD")

two_bars_plot(prim_value = "num_hi_TMB_Primary",
              meta_value = "num_hi_TMB_Metastasis",
              annot = False,
              y_positions = y_positions,
              y_lim = y_lim,
              ax = ax_tmb_hi,
              xlabel = "Fraction",
              x_max = 1,
              h=0.4,
              title = r"TMB$^{hi}$")

actionability_plot(y_positions=y_positions,
                   y_lim = y_lim,
                   ax=ax_action, h=0.4, p=0.04)

violin_plot(data = fga_data,
            y_positions = y_positions,
            y_lim = y_lim,
            ax = ax_fga,
            xlabel = "%",
            x_max= 100,
            title = "FGA")

violin_plot(data = tmb_data,
```

```
            y_positions = y_positions,
            y_lim = y_lim,
            ax = ax_tmb,
            xlabel = "mut/Mb",
            logscale=True,
            title = "TMB")

subtypes = fig2b_table.index.to_list()
for ax, st in zip(ax_heatmaps, subtypes):
    cna_heatmap_plot(st, ax)

# plot legends
plt.rcParams["legend.handlelength"] = 1
plt.rcParams["legend.labelspacing"] = 0.25
plt.rcParams["legend.frameon"] = False
plt.rcParams["legend.title_fontsize"] = 7
plt.rcParams["legend.fontsize"] = 7
plt.rcParams["legend.loc"] = "upper left"

ax_action.legend(["L1", "L2", "L3A", "L3B", "L4", "Onc.", "VUS", "None"],
                 title = "Actionabl.", bbox_to_anchor = (-14,-0.01), ncol = 4,␣
 ↪columnspacing = 0.5, handletextpad = 0.5)

norm = plt.Normalize(vmin =-0.26, vmax =0.26)
sm = plt.cm.ScalarMappable(cmap= sns.color_palette("RdBu_r", as_cmap=True),␣
 ↪norm=norm)
sm.set_array([])

cax = fig.add_axes([0.7, -0.05, 0.2, 0.01])
cbar = plt.colorbar(sm, cax=cax, orientation = 'horizontal')
cbar.ax.set_title(r'Loss $\longleftrightarrow$ Gain', fontsize = 8)
cbar.ax.tick_params(labelsize=7);

ax_heatmaps[0].set_xticks(np.arange(len(heatmap_top))+0.5)
ax_heatmaps[0].set_xticklabels(heatmap_top, rotation = 90, fontsize = 5)
ax_heatmaps[0].xaxis.tick_top() # x axis on top
ax_heatmaps[0].xaxis.set_label_position('top')
ax_heatmaps[0].set_yticks([])
ax_heatmaps[0].tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
ax_heatmaps[0].set_axis_on()

fig.savefig("fig2b.png", bbox_inches = "tight")
```


No. patients
FGA
WGD
TMB
TMB^hi Actionability
Small Cell Lung Cancer (314)
Lung Neuroendocrine (109)
Lung Squamous Cell Carcinoma (537)
Lung Adenocarcinoma (4064)
Pleural Mesothelioma (244)
Uterine Carcinosarcoma (192)
Ovarian High-Grade Serous (997)
Uterine Serous (288)
Cervical Squamous Cell (103)
Ovarian Low-Grade Serous (89)
Ovarian Clear Cell (97)
Uterine Endometrioid (567)
Uterine Hypermutated (268)
Breast Ductal Triple Negative (341)
Breast Ductal HR+HER2+ (256)
Breast Ductal HR-HER2+ (115)
Breast Ductal HR+HER2- (1524)
Breast Lobular HR+ (373)
Pancreatic Neuroendocrine (211)
Hepatocellular Carcinoma (205)
Pancreatic Adenocarcinoma (1779)
Cholangio Intrahepatic (407)
Cholangio Extrahepatic (119)
Gallbladder Cancer (171)
Melanoma Mucosal (175)
Melanoma Cutaneous (864)
Melanoma Uveal (103)
Cutaneous Squamous Cell (105)
Sarcoma Leiomyo (291)
Sarcoma UPS/MXF (203)
Sarcoma Lipo (228)
Esophageal Adenocarinoma (542)
Gastrointestinal Stromal (403)
Colorectal MSS (3197)
Anal Squamous Cell (87)
Gastrointestinal Neuroendocrine (150)
Small Bowel cancer (94)
Stomach Adenocarinoma (320)
Colorectal Hypermutated (351)
Appendiceal Adenocarcinoma (202)
Prostate Adenocarcinoma (2172)
Testicular Non-Seminoma (240)
Testicular Seminoma (97)
Upper Tract Urothelial (197)
Bladder Urothelial (961)
Renal Clear Cell (421)
Head and Neck Squamous (412)
Adenoid Cystic Carcinoma (152)
Thyroid Poorly Differentiated (131)
Thyroid Papillary (307)
Actionabl.
L1
L2
L3A
L3B
L4
Onc.
VUS
#
%
Fraction
mut/Mb
Fraction
Fraction
Loss ⟷ Gain
−0.2
0.0
0.2

[ ]: